IBM

Application Program

GH20-0586-0

System/360 Scientific Subroutine Package (PL/I)

(360A-CM-07X)

Program Description and Operations Manual

The System/360 Scientific Subroutine Package (SSP) (PL/I) is a collection of mathematical and statistical subroutines (or procedures) written in the PL/I language. It provides the PL/I user with most of the basic capabilities in earlier FORTRAN versions of SSP/360. It also has the same basic characteristics as the FORTRAN versions, in that it consists of input/output-free computational building blocks, written completely in PL/I, which may be combined with a user's input, output, or computational routines as needed. The package may be applied to the solution of many problems in industry, science, and engineering.

This manual contains sufficient information to permit the reader to understand and use all of the subroutines in the Scientific Subroutine Package.

Note: This programming package has been developed with the cooperation and assistance of IBM Germany and IBM France.

First Edition (January 1968)

Significant changes or additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

This edition applies to Version 1, Modification Level 0 of System/360 Scientific Subroutine Package (PL/I) (360A-CM-07X) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the specifications herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (N20-0360) 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, 1133 Westchester Avenue, White Plains, New York 10604.

CONTENTS

INTRODUCTION

The Scientific Subroutine Package (SSP) for Operating System/360 PL/I is a set of basic computational subroutines intended to help the user develop his own PL/I program library. The user may supplement or modify the subroutines to meet his needs. This package 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. As with all tools, the user should understand their capabilities and their application to his functional requirements before deciding to use them.

AREAS OF APPLICATION

Individual subroutines or a combination of them can be used for the general areas listed here.

Mathematics

- Matrix operations
  - Elementary
  - Linear equations
  - Eigenvalues
- Polynomial operations
  - Orthogonal polynomials
  - Polynomial economization
  - Polynomial roots
- Numerical quadrature
  - Tabulated functions
  - Nontabulated functions
- Numerical differentiation
  - Tabulated functions
  - Nontabulated functions
- Interpolation of tabulated functions
- Approximation of tabulated functions
- Smoothing of tabulated functions
- Roots and extrema of functions
- Systems of ordinary differential equations
- Special mathematical functions

Statistics

- Data screening and analysis
- Elementary statistics
- Correlation and regression analysis
  - Correlation
  - Multiple linear regression
  - Stepwise multiple regression
  - Canonical correlation
- Analysis of variance
- Discriminant analysis
- Principal components analysis
- Nonparametric statistics
- Distribution functions

IBM REFERENCE MATERIAL

System/360 Scientific Subroutine Package (360A-CM-03X) Version III Programmer's Manual (H20-0205)

IBM System/360 Operating System PL/I (F) Reference Manual (C28-8201)

IBM System/360 Operating System PL/I (F) Programmer's Guide (C28-6594)

Preface to PL/I Programming in Scientific Computing (E20-0312)

CHARACTERISTICS

Some of the characteristics of SSP/360 (PL/I) are as follows:

- All subroutines are free of input/output statements.
- All subroutines are written in OS/360 PL/I (F).
- Most of the subroutines provide a double-precision option.
- The use of certain subroutines (or groups of them) is illustrated in the program documentation by sample main programs with input/output.
- All subroutines are documented uniformly.

An example of a sample main program that uses several of the subroutines is the statistical function called Principal Components Analysis (FACT).* It uses five separate subroutine capabilities, as follows:

- Computation of means, standard deviations, and correlation matrix (CORR)
- Computation of eigenvalues and eigenvectors of the correlation matrix (MSDU)
- Selection of eigenvalues (TRAC)
- Computation of factor matrix (LOAD)
- Varimax rotation of the factor matrix (VRMX)

This is one of the sample main programs included in the program documentation.

*This program performs the same functions as the program that was called Factor Analysis in the FORTRAN versions of SSP. The name Principal Components Analysis more aptly describes the function of this program than the name Factor Analysis. For a discussion of the distinction between Factor Analysis and Principal Components Analysis see Section 2.2 of 1130 Statistical System (1130-CA-06X) User's Manual (H20-0333).

REQUIRED SYSTEMS

Programming Systems

The subroutines are written in the PL/I language, using the 48-character set and the facilities provided by the PL/I (F) compiler, which functions under Operating System/360.

Machine Configuration

A minimum requirement is a System/360 suitable for the OS/360 PL/I (F) compiler. The machine configuration required for any given problem depends on 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.

OVERALL RULES OF USAGE

GENERAL RULES

All subroutines in SSP are entered by means of the standard PL/I CALL statement. The 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, the input/output and other operations necessary for the total solution of his problem. He must also define by DECLARE 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 used like any user-supplied subroutine. All of the normal rules of PL/I concerning subroutines must therefore be followed. Note that the subroutines have been written using the 48-character set, so the programmer should be familiar with its use.

All variables in the calling program must be declared with the proper attributes. Those variables appearing as parameters in the call statement of the calling program should not have attributes conflicting with those of the called 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. When the argument is an array, the address and size of the array are transmitted to the called subroutine.

The arguments in a CALL statement should agree in order, number, and type with the corresponding arguments in the subroutine. In SSP, all arguments in a CALL statement must be variable names. Constants are not acceptable. For example, if the user wishes to invert a matrix A, which is 10 by 10, using the SSP subroutine MINV, and if the constant for testing the condition of the matrix is 10^{-8}, these constants must be defined as variables before calling MINV, as illustrated below:

```
N = 10, .

CON = 1.0 E - 8, .

CALL MINV (A, N, D, CON), .
```

where D is the determinant.

Some of the subroutines in SSP require the name of a user function subprogram or a PL/I-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 be declared as ENTRY in the user's calling program.

For example, the SSP routine SBST calls a user-supplied subroutine. The user must, therefore, prepare a subroutine, with the proper argument list, to perform the desired tasks. He must declare the name of this subroutine as ENTRY in his calling program and supply the name of that subroutine to SBST as the appropriate parameter in his CALL statement to subroutine SBST. The subroutine SBST need not be modified by the user. The dummy argument B in the subroutine SBST is replaced by the user's subroutine name at execution time.

The following illustrates these procedures:

SSP Subroutine SBST (need not be altered)

```
SBST..
   PROCEDURE (A, C, R, B, S, NO, NV, NC),.
   DECLARE
         B ENTRY,.
      .
      .
      .
   CALL B (R, TR), .
      .
      .
      .
   RETURN, .
   END, .
```

User's Calling Program

```
USER. .
   PROCEDURE OPTIONS (MAIN),.
   DECLARE
         BOOL ENTRY, .
      .
      .
      .
   CALL SBST (A, C, R, BOOL, S, NO, NX,
   NC),.
      .
      .
      .
   RETURN, .
   END, .
```

User's Function Subprogram

```
BOOL..
   PROCEDURE (R, T), .
        .
        .
        .
   RETURN, .
   END, .
```

ERROR CODES

In the Scientific Subroutine Package most of the subroutines use an error indicator to warn the user that a certain condition exists. The user, in his calling program, should check the error indicator when returning from a called program. If the user wishes to use the error indicator as an aid, he should, in his calling program, declare ERROR EXTERNAL CHARACTER(1). In this way he has available in the calling program the value of the error indicator (ERROR).

If, in using a subroutine, an error is detected, some of the output areas may contain invalid data. Generally, however, output areas are set to appropriate values (for example, zero or $\pm 10^{75}$).

MATRIX OPERATIONS

Special consideration must be given to the subroutines that perform matrix operations. These subroutines have two characteristics that affect the size and 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 the asterisk notation. Under this approach, where a called subroutine needs to declare an array of the same dimensions as a calling program, the dimension specifications are replaced by asterisks. Thus, the user does not need to modify the subroutines so long as he has declared adequate dimensions for arrays in the calling program or main program.

One way to ensure that arrays have adequate dimensions for various problems is to declare them with variable notations. For example, if matrix R contains intercorrelation coefficients among M variables, the DECLARE statement appears as follows:

```
DECLARE R(M,M), .
```

If M is 10, then 100 locations will be allocated for matrix R.

If M is 20, then 400 locations will be allocated automatically.

Storage Compression

When working with symmetric matrices it is often advantageous to use a compressed (vector) storage form. This means that only the upper or lower triangular part of the matrix need be stored, which for an N by N matrix reduces the core requirements from N^2 locations to $N(N+1)/2$ locations. A subroutine, MSCS, is provided in this package which stores a symmetric matrix in compressed form and at the same time tests the matrix for symmetry. The element stored is the average of each pair of symmetric elements of an n by n matrix Q, i.e.,

$$S_{ik} = \frac{Q_{ik} + Q_{ki}}{2} \qquad \begin{array}{l} i = 1, \ldots, n. \\ k = 1, \ldots, i. \end{array}$$

At the same time the difference $Q_{ik} - Q_{ki}$ is tested against a user-supplied tolerance. If this test fails, an ERROR indication is given but in any case the results S_{ik} are supplied in the vector form:

$$S_{11}, S_{21}, S_{22}, S_{31}, S_{32}, S_{33} \ldots S_{nn}$$

Another subroutine, MSCG, is provided which converts this vector (compressed) form back to the general two-dimensional form.

Some of the subroutines of SSP -- for example, MMSS and MAGS -- accept input in this compressed form.

DOUBLE 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. The user may, therefore, wish to use double-precision versions of these subroutines. Most of the SSP/360 (PL/I) subroutines provide a double-precision option. PL/I double-precision statements have been included in each of these subroutines in the form of a comments card. The double-precision version of the subroutine can be obtained by removing/* from cc 3 and 4 of the double-precision statement card(s) and by removing the corresponding single-precision cards (or making them comments cards) before compilation. The use of double-precision subroutines requires a detailed knowledge of the PL/I rules concerning double precision. Two of the more basic rules are as follows:

1. Any real variable, vector, or array name contained in the argument list of a CALL to a double-precision subroutine must be declared as double precision in the calling program.

2. Any user-supplied function named in the CALL statement for a double-precision SSP subroutine must be programmed as a double-precision function.

FORMAT OF THE DOCUMENTATION

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

SUBROUTINE DESCRIPTIONS

Subroutines and sample program guides, both categorical and alphabetic, designed to help locate particular subroutines are given in the pages that follow.

The subroutine descriptions, in general, consist of purpose, usage, remarks, method, mathematical background, programming considerations, and a program listing. References to books and periodicals will be found under the method section of the description. The mathematical description pages do not, in all cases, indicate the derivation of the mathematics. They are intended to indicate what mathematical operations are actually being performed in the subroutines.

SAMPLE PROGRAM DESCRIPTIONS

A sample program, in general, consists of a description of the problem, program, input, output, program modification, operating instructions, error messages, timing, machine listing of the program, sample input data, and output results. In some cases (for example, as a part of developing the data screening sample program) a special sample subroutine has been implemented that may prove useful to the programmer. One such subroutine, called HIST, prints a histogram of frequencies. The listing of these subroutines is included after the sample program documentation in this manual.

Instructions for modifying the sample programs for different data formats are included in the documentation. In addition, those sample programs that illustrate potentially double-precision subroutines include double-precision statements in the form of comment cards. These comment cards are contained in the sample program source decks.

OPERATING NOTES

It is recommended that those SSP subroutines that will be frequently used in an installation be compiled and that the relocatable programs be placed on the PL/I systems residence device. In the case of Operating System/360, this will be the PL/I library portion of the system disk pack. Information on the method for updating the system to include user-supplied subroutines appears in the appropriate PL/I programmer's guide. SSP subroutines are handled in the same manner as user-supplied subroutines. If the subroutines are not placed in the PL/I library, those required by a particular program will have to be included in that program each time it is run. As noted earlier, the subroutines have been written using the 48-character set.

CATEGORICAL GUIDE TO SUBROUTINES AND SAMPLE PROGRAMS

MATHEMATICS

Matrix Operations

Elementary Operations

Linear Equations and Related Topics

Eigenvalues and Related Topics

Polynomial Operations

Numerical Quadrature

Quadrature of Tabulated Functions

Quadrature of Nontabulated Functions

Numerical Differentiation

Differentiation of Tabulated Functions

Differentiation of Nontabulated Functions

Interpolation of Tabulated Functions

Approximation of Tabulated Functions

Smoothing of Tabulated Functions

Page

Roots and Extrema of Functions

Systems of Ordinary Differential Equations

Special Mathematical Functions

STATISTICS

Data Screening and Analysis

Elementary Statistics

Correlation and Regression Analysis

Analysis of Variance

Discriminant Analysis

Principal Components Analysis

Nonparametric Statistics

Distribution Functions

GUIDE TO SAMPLE PROGRAMS

Data Screening

Multiple Linear Regression

Page

Stepwise Multiple Regression

Canonical Correlation

Analysis of Variance

Discriminant Analysis

Page

Principal Components Analysis

Kolmogorov-Smirnov Test

Triple Exponential Smoothing

Allocation of Overhead Costs

ALPHABETIC GUIDE TO SUBROUTINES AND SAMPLE PROGRAMS, WITH STORAGE REQUIREMENTS

The following table lists the number of bytes of storage for the program control section required by each of the subroutines in the Scientific Subroutine Package. The storage requirements were obtained by using Version 4 of PL/I and Release 16 of OS. The use of other versions and releases may cause deviations from these figures.

The double-precision version storage requirements of the subroutines in the Scientific Subroutine Package are included in parentheses.

Name	Math. Description Page Number	Storage Required Bytes
ABST	183	610
ACFM	126	2,826 (2,696)
ACFE	126	
AHIM	122	2,946 (2,950)
AHIE	122	
ALIM	118	2,306 (2,310)
ALIE	118	
ANOV	274	4,482
APC1	140	1,766 (1,766)
APC2	140	
APLL	139	986 (986)
ASN	143	1,902 (1,874)
AVAR	206	4,174 (4,174)
BDTR	240	3,830
BOOL	259	266
BOUN	182	1,102
CANC	204	4,718 (4,718)
CANO	270	5,478
CDTR	243	3,962
CEL1	172	858 (854)
CEL2	172	
CHSQ	224	3,882
CORR	194	4,352 (4,408)
COST	294	3,206
DACR	255	4,294
DAT1	259	1,098
DAT2	265	1,098
DAT3	277	850
DERE	167	2,762 (2,738)
DET3	108	658 (658)
DET5	110	890 (890)
DFEC	112	1,142 (1,142)
DFEO	115	1,118 (1,118)
DGT3	107	894 (894)
DMTX	209	2,498 (2,510)
DSCR	210	3,090 (3,110)
ELI1	174	1,458 (1,454)
ELI2	174	
EXPN	291	2,430
EXSM	152	1,030
FACT	281	7,116
FFT	129	3,166 (3,166)
FFTM	134	4,040 (4,040)
FMFP	153	4,174 (4,040)
HIST	259	2,674
HTES	238	1,122
IDT1	265	614
IDT2	270	614
JELF	177	1,270 (1,270)
KLMO	218	2,010
KLM2	221	1,998
KOLM	286	6,828
KRNK	227	2,010
LGAM	180	750
LOAD	214	666 (666)
MAGS	14	638 (638)
MATE	56	1,706
MATU	58	1,918
MDLG	39	1,314
MDLS	35	1,426 (1,414)
MDRS	35	
MDSB	37	1,202 (1,186)
MDSC	277	6,482
MEAT	61	5,638
MEBS	66	1,066
MEST	63	1,890
MFG	23	1,882 (1,858)
MFGR	29	2,730 (2,714)
MFS	25	886 (874)
MFSB	27	1,158 (1,142)
MGB1	49	3,562 (3,550)
MGB2	49	
MGDU	71	2,274 (2,274)
MIG	40	1,894 (1,858)
MINV	44	3,014 (3,014)
MIS	42	1,198 (1,182)
MLSQ	45	3,622 (3,558)
MLTR	197	2,098 (2,098)
MMGG	15	630 (622)
MMGS	17	1,062 (1,058)
MMGT	18	858 (846)
MMSS	16	718 (710)
MOMN	191	2,078
MPRM	19	1,078 (1,078)
MPIT	21	730
MSCG	14	474 (474)
MSCS	13	626 (626)
MSDU	69	3,538 (3,538)
MSTU	59	2,426
MTPI	20	674

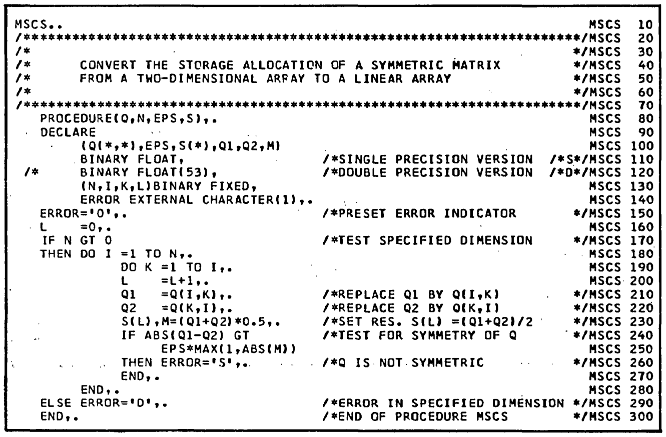
Name	Math. Description Page Number	Storage Required Bytes
MVAT	72	5,782
MVEB	76	1,294
MVST	67	3,254
MVSU	74	1,182
MVUB	75	1,518
NDTI	246	834
NDTR	239	450
ORDR	196	1,238 (1,238)
PEC	81	2,082 (2,090)
PTC	81	
POST	86	1,322 (1,322)
POSV	78	798 (790)
POV	77	722 (714)
PRTC	87	2,686 (2,718)
QA2	105	362 (354)
QA4	105	510 (490)
QA8	105	398 (398)
QA12	105	402 (402)
QA16	105	402 (402)
QA24	105	398 (394)
QATR	97	1,318 (1,318)
QG2	99	422 (422)
QG4	99	574 (554)
QG8	99	534 (526)
QG16	99	538 (530)
QG24	99	538 (530)
QG32	99	538 (530)
QG48	99	530 (522)
QH2	103	346 (342)
QH4	103	474 (466)
QH8	103	454 (450)
QH16	103	458 (454)
QH24	103	458 (454)
QH32	103	458 (454)
QH48	103	450 (446)
QHFG	94	1,178 (1,178)
QHFE	94	
QHSG	94	
QHSE	94	
QL2	101	362 (354)
QL4	101	510 (490)
QL8	101	398 (398)
QL12	101	402 (402)
QL16	101	402 (402)
QL24	101	398 (394)
QSF	93	710 (710)
QTFG	92	702 (702)
QTFE	92	
QTST	229	1,462
RANK	230	962
REGR	260	7,930
RTF	159	1,878 (1,882)
RTFD	163	1,762 (1,762)
SBST	184	1,562
SE13	147	1,118 (1,118)
SG13		
SE15	149	730 (730)
SE35	150	774 (774)
SMIR	223	710
SRNK	231	1,558
SOUT	270	3,458
STEP	265	5,494
STRG	200	4,914 (4,950)
SUBM	190	790
TAB1	185	2,642
TAB2	187	4,894
TALY	181	2,090
TIE	233	926
TRAC	213	818 (818)
TTST	192	2,562
TWAV	234	1,562
UTST	235	1,302
VRMX	215	3,970 (3,852)
WTST	236	1,986

MATHEMATICS

Matrix Operations

Elementary Operations

- Subroutine MSCS



```
MSCS..                                                                MSCS
/*********************************************************************/MSCS
/*                                                                  */MSCS
/*     CONVERT THE STORAGE ALLOCATION OF A SYMMETRIC MATRIX         */MSCS
/*     FROM A TWO-DIMENSIONAL ARRAY TO A LINEAR ARRAY               */MSCS
/*                                                                  */MSCS
/*********************************************************************/MSCS
   PROCEDURE(Q,N,EPS,S),.                                             MSCS
   DECLARE                                                            MSCS
        (Q(*,*),EPS,S(*),Q1,Q2,M)                                     MSCS
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/MSCS
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/MSCS
        (N,I,K,L)BINARY FIXED,                                        MSCS
        ERROR EXTERNAL CHARACTER(1),.                                 MSCS
   ERROR='0',.                        /*PRESET ERROR INDICATOR      */MSCS
   L    =0,.                                                          MSCS
   IF N GT 0                          /*TEST SPECIFIED DIMENSION    */MSCS
   THEN DO I =1 TO N,.                                                MSCS
             DO K =1 TO I,.                                           MSCS
             L    =L+1,.                                              MSCS
             Q1   =Q(I,K),.           /*REPLACE Q1 BY Q(I,K)        */MSCS
             Q2   =Q(K,I),.           /*REPLACE Q2 BY Q(K,I)        */MSCS
             S(L),M=(Q1+Q2)*0.5,.     /*SET RES. S(L) =(Q1+Q2)/2    */MSCS
             IF ABS(Q1-Q2) GT         /*TEST FOR SYMMETRY OF Q      */MSCS
                  EPS*MAX(1,ABS(M))                                   MSCS
             THEN ERROR='S',.         /*Q IS NOT SYMMETRIC          */MSCS
             END,.                                                    MSCS
        END,.                                                         MSCS
   ELSE ERROR='D',.                   /*ERROR IN SPECIFIED DIMENSION */MSCS
   END,.                              /*END OF PROCEDURE MSCS       */MSCS
```

Purpose:

MSCS compresses the storage allocation of a symmetric two-dimensional matrix to a one-dimensional array.

Usage:

CALL MSCS (Q, N, EPS, S);

Q(N, N) -	BINARY FLOAT [(53)] Given N by N symmetric matrix.
N -	BINARY FIXED Given order of matrices Q and S.
EPS -	BINARY FLOAT [(53)] Given relative tolerance for test on symmetry.
S(N*(N+1)/2) -	BINARY FLOAT [(53)] Resultant symmetric matrix in one-dimensional compressed form.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='D' means N is less than 1.

ERROR='S' means given matrix Q does not pass the specified symmetry test. Nonetheless, all of the elements S_{ik} are computed as shown below and stored in S.

Method:

$$S_{ik} = \frac{Q_{ik} + Q_{ki}}{2} \quad \text{for } i=1,2,\ldots,n \quad k=1,\ldots,i$$

Symmetry-test:

$Q_{ik} - Q_{ki}$ must be absolutely less than

$$\text{Max}\ (1, \frac{|Q_{ki} + Q_{ik}|}{2}) * \text{EPS}$$

Subroutine MSCG

```
MSCG..                                                                  MSCG  10
/**********************************************************************/MSCG  2C
/*                                                                    */MSCG  30
/*      CONVERT THE STORAGE ALLOCATION OF A SYMMETRIC MATRIX          */MSCG  40
/*      FROM A LINEAR ARRAY TO A TWO-DIMENSIONAL ARRAY                */MSCG  50
/*                                                                    */MSCG  60
/**********************************************************************/MSCG  70
   PROCEDURE(S,N,Q),.                                                   MSCG  8C
   DECLARE                                                              MSCG  90
        (S(*),Q(*,*))                                                   MSCG 100
        BINARY FLOAT,                      /*SINGLE PRECISION VERSION /*S*/MSCG 110
 /*     BINARY FLOAT(53),                  /*DOUBLE PRECISION VERSION /*D*/MSCG 120
        (N,I,K,L)BINARY FIXED,.                                         MSCG 130
   L    =0,.                                                            MSCG 140
   IF N GT 0                               /*TEST SPECIFIED DIMENSION   */MSCG 150
   THEN DO I =1 TO N,.                                                  MSCG 160
             DO K =1 TO I,.                                             MSCG 170
             L    =L+1,.                                                MSCG 180
             Q(I,K),Q(K,I)=S(L),.          /*STORE Q(I,K) AND Q(K,I)    */MSCG 190
             END,.                                                      MSCG 200
        END,.                                                           MSCG 210
   END,.                                   /*END OF PROCEDURE MSCG      */MSCG 22C
```

Purpose:

MSCG expands the compressed one-dimensional storage allocation of a symmetric matrix to general two-dimensional form.

Usage:

CALL MSCG (S, N, Q);

S(N*(N+1)/2) - BINARY FLOAT [(53)]
Given one-dimensional array representing a symmetric N by N matrix in compressed form.

N - BINARY FIXED
Given order of matrices S and Q.

Q(N, N) - BINARY FLOAT [(53)]
Resultant two-dimensional general representation of given symmetric matrix S.

Remarks:

Operation is bypassed in case of a nonpositive value of N. The elements of given S are assumed to be stored in compressed form -- that is:

$$(S_{11}, S_{21}, S_{22}, S_{31}, S_{32}, S_{32}, \ldots, S_{n1}, \ldots, S_{nn})$$

Method:

For the elements of resultant Q:

$$Q_{ik} = Q_{ki} = S_{ik} \text{ for } \begin{array}{l} i = 1, 2, \ldots, n \\ k = 1, 2, \ldots, i \end{array}$$

Subroutine MAGS

```
MAGS..                                                                  MAGS  10
/**********************************************************************/MAGS  20
/*                                                                    */MAGS  30
/*      ADD OR SUBTRACT A SQUARE AND A SYMMETRIC MATRIX               */MAGS  40
/*                                                                    */MAGS  5C
/**********************************************************************/MAGS  60
   PROCEDURE(A,B,N,OPT,C),.                                             MAGS  7C
   DECLARE                                                              MAGS  80
        (A(*,*),B(*),C(*,*),AL,BL)                                      MAGS  90
        BINARY FLOAT,                      /*SINGLE PRECISION VERSION /*S*/MAGS 100
 /*     BINARY FLOAT(53),                  /*DOUBLE PRECISION VERSION /*D*/MAGS 110
        (N,I,K,L,LI)BINARY FIXED,                                       MAGS 120
        OPT CHARACTER(1),.                                              MAGS 130
   IF N GT 0                               /*IS N GREATER THAN ZERO     */MAGS 14C
   THEN DO,.                                                            MAGS 150
        LI,I =1,.                                                       MAGS 160
NEXTI..                                                                 MAGS 170
        L    =LI,.                                                      MAGS 180
        K    =1,.                                                       MAGS 19C
NEXTK..                                                                 MAGS 200
        AL   =A(I,K),.                     /*REPLACE AL BY A(I,K)       */MAGS 21C
        BL   =B(L),.                       /*SET BL CORRESPONDING TO AL */MAGS 220
        IF K LT I                                                       MAGS 23C
        THEN L    =L+1,.                                                MAGS 240
        ELSE L    =L+K,.                                                MAGS 250
        IF OPT='2'                         /*SHOULD A-B BE CALCULATED   */MAGS 26C
        THEN BL   =-BL,.                   /*THEN CONVERT SIGN OF BL    */MAGS 270
        ELSE IF OPT='3'                    /*SHOULD B-A BE CALCULATED   */MAGS 280
        THEN AL   =-AL,.                   /*THEN CONVERT SIGN OF AL    */MAGS 290
        C(I,K)=AL+BL,.                     /*SET RESULTANT C(I,K) TO AL+BL*/MAGS 300
        IF K LT N                                                       MAGS 310
        THEN DO,.                          /*INCREMENT K                */MAGS 320
             K    =K+1,.                                                MAGS 330
             GO TO NEXTK,.                                              MAGS 340
             END,.                                                      MAGS 350
        ELSE IF I LT N                                                  MAGS 360
        THEN DO,.                          /*INCREMENT I                */MAGS 370
             LI   =LI+I,.                                               MAGS 380
             I    =I+1,.                                                MAGS 390
             GO TO NEXTI,.                                              MAGS 400
             END,.                                                      MAGS 410
        END,.                                                           MAGS 420
   END,.                                   /*END OF PROCEDURE MAGS      */MAGS 430
```

Purpose:

MAGS computes $C = A + B$ if OPT = '1'
$C = A - B$ if OPT = '2'
$C = B - A$ if OPT = '3'

for given matrices A and B which are general and symmetric respectively.

Usage:

CALL MAGS (A, B, N, OPT, C);

A(N, N) - BINARY FLOAT [(53)]
Given general N by N matrix.

B(N*(N+1)/2) - BINARY FLOAT [(53)]
Given one-dimensional array containing the lower triangular part of symmetric matrix B stored rowwise in compressed form.

N - BINARY FIXED
Given order of matrices A, B and C.

OPT - CHARACTER(1)
Given option for selection of operation.

C(N, N) - BINARY FLOAT [(53)]
Resultant general N by N matrix, which may be overlaid with A.

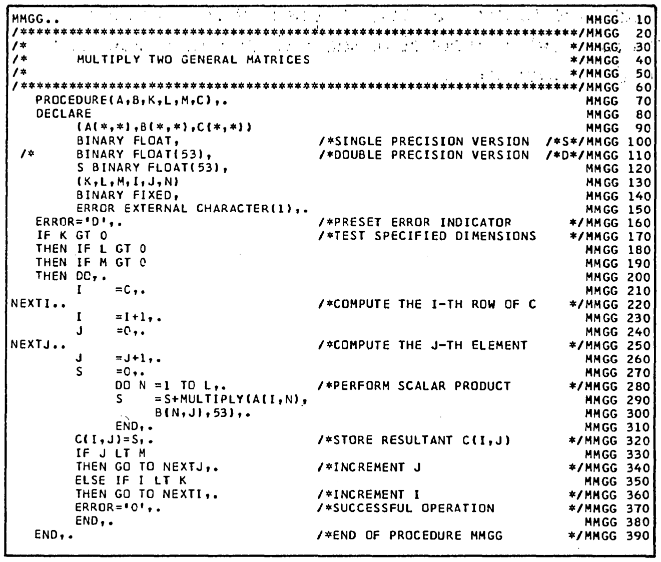
Remarks:

Operation is bypassed in case of a nonpositive value of N. A value of OPT different from '2' and '3' is treated as if it were '1'.

Method:

The sum or difference of matrices A and B is calculated elementwise. The elements of the symmetric matrix B are accessed only once.

- Subroutine MMGG



```
MMGG..
/*********************************************************************/
/*                                                                   */
/*      MULTIPLY TWO GENERAL MATRICES                                */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(A,B,K,L,M,C),.
   DECLARE
        (A(*,*),B(*,*),C(*,*))
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        S BINARY FLOAT(53),
        (K,L,M,I,J,N)
        BINARY FIXED,
        ERROR EXTERNAL CHARACTER(1),.
   ERROR='D',.                        /*PRESET ERROR INDICATOR       */
   IF K GT 0                          /*TEST SPECIFIED DIMENSIONS    */
   THEN IF L GT 0
   THEN IF M GT 0
   THEN DO,.
        I    =C,.
NEXTI..                               /*COMPUTE THE I-TH ROW OF C    */
        I    =I+1,.
        J    =0,.
NEXTJ..                               /*COMPUTE THE J-TH ELEMENT     */
        J    =J+1,.
        S    =0,.
             DO N =1 TO L,.           /*PERFORM SCALAR PRODUCT       */
             S    =S+MULTIPLY(A(I,N),
                  B(N,J),53),.
             END,.
        C(I,J)=S,.                    /*STORE RESULTANT C(I,J)       */
        IF J LT M
        THEN GO TO NEXTJ,.            /*INCREMENT J                  */
        ELSE IF I LT K
        THEN GO TO NEXTI,.            /*INCREMENT I                  */
        ERROR='0',.                   /*SUCCESSFUL OPERATION         */
        END,.
   END,.                              /*END OF PROCEDURE MMGG        */
```

Purpose:

MMGG computes the standard matrix product C = A • B.

Usage:

CALL MMGG (A, B, K, L, M, C);

A(K, L) - BINARY FLOAT [(53)]
Given K by L matrix A (left-hand factor).

B(L, M) - BINARY FLOAT [(53)]
Given L by M matrix B (right-hand factor).

K - BINARY FIXED
Given row dimension of A and C.

L - BINARY FIXED
Given column dimension of A and row dimension of B.

M - BINARY FIXED
Given column dimension of B and C.

C(K, M) - BINARY FLOAT [(53)]
Resultant K by M product matrix.

Remarks:

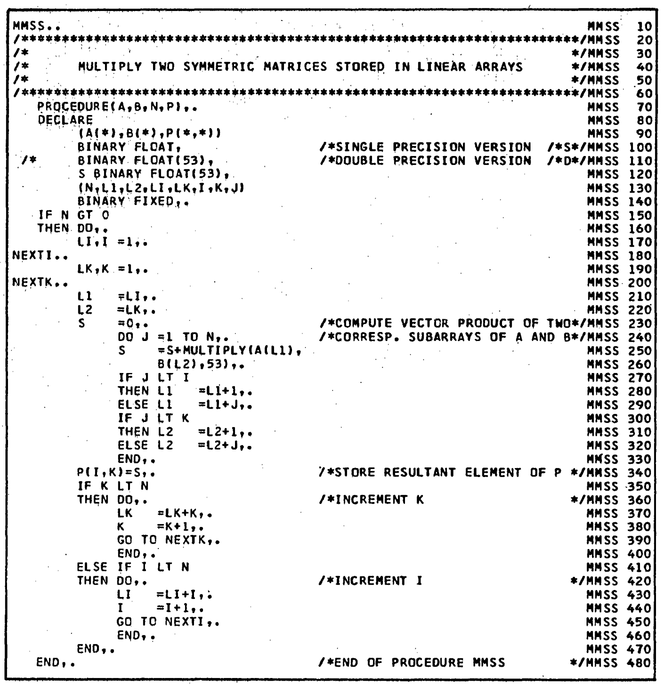
If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='D' means errors in specified dimensions K, L, M. Accumulation of scalar products is performed in double-precision arithmetic. C must be different from A and B.

Method:

Standard multiplication means that the element C_{ik} is the scalar product of the i-th row of A with the k-th column of B.

- Subroutine MMSS



```
MMSS..
/**********************************************************************/
/*                                                                    */
/*      MULTIPLY TWO SYMMETRIC MATRICES STORED IN LINEAR ARRAYS       */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,B,N,P),.
   DECLARE
        (A(*),B(*),P(*,*))
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        S BINARY FLOAT(53),
        (N,L1,L2,LI,LK,I,K,J)
        BINARY FIXED,.
   IF N GT 0
   THEN DO,.
        LI,I =1,.
NEXTI..
        LK,K =1,.
NEXTK..
        L1   =LI,.
        L2   =LK,.
        S    =0,.                     /*COMPUTE VECTOR PRODUCT OF TWO*/
             DO J =1 TO N,.           /*CORRESP. SUBARRAYS OF A AND B*/
             S    =S+MULTIPLY(A(L1),
                  B(L2),53),.
             IF J LT I
             THEN L1   =L1+1,.
             ELSE L1   =L1+J,.
             IF J LT K
             THEN L2   =L2+1,.
             ELSE L2   =L2+J,.
             END,.
        P(I,K)=S,.                    /*STORE RESULTANT ELEMENT OF P */
        IF K LT N
        THEN DO,.                     /*INCREMENT K                  */
             LK   =LK+K,.
             K    =K+1,.
             GO TO NEXTK,.
             END,.
        ELSE IF I LT N
        THEN DO,.                     /*INCREMENT I                  */
             LI   =LI+I,.
             I    =I+1,.
             GO TO NEXTI,.
             END,.
        END,.
   END,.                              /*END OF PROCEDURE MMSS        */
```

Purpose:

MMSS computes the standard product P = A · B of two symmetric matrices.

Usage:

CALL MMSS (A, B, N, P);

A(N*(N+1)/2) - BINARY FLOAT [(53)]
Given symmetric N by N matrix, stored in compressed form (left-hand factor).

B(N*(N+1)/2) - BINARY FLOAT [(53)]
Given symmetric N by N matrix, stored in compressed form (right-hand factor).

N - BINARY FIXED
Given order of matrices A, B, P.

P(N, N) - BINARY FLOAT [(53)]
Resultant N by N general product matrix.

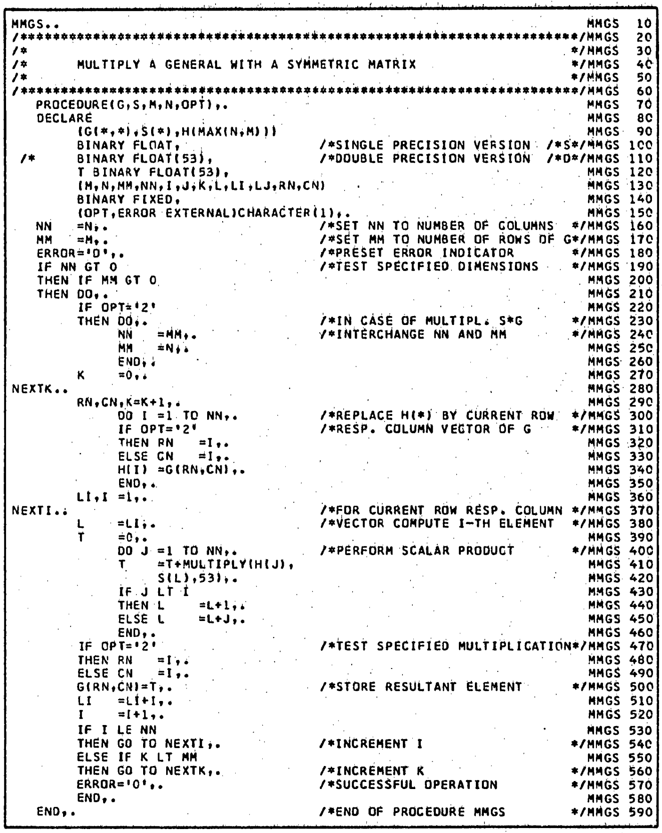
Remarks:

Operation is bypassed in case of a nonpositive value of N. The symmetric matrices A and B must be stored in compressed form. Accumulation of scalar products is performed in double-precision arithmetic.

Method:

Standard multiplication means that the element P_{ik} is the scalar product of the i-th row of A with the k-th column of B.

- <u>Subroutine</u> MMGS



```
MMGS..
/************************************************************************/
/*                                                                      */
/*      MULTIPLY A GENERAL WITH A SYMMETRIC MATRIX                      */
/*                                                                      */
/************************************************************************/
   PROCEDURE(G,S,M,N,OPT),.
   DECLARE
        (G(*,*),S(*),H(MAX(N,M)))
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        T BINARY FLOAT(53),
        (M,N,MM,NN,I,J,K,L,LI,LJ,RN,CN)
        BINARY FIXED,
        (OPT,ERROR EXTERNAL)CHARACTER(1),.
   NN   =N,.                           /*SET NN TO NUMBER OF COLUMNS  */
   MM   =M,.                           /*SET MM TO NUMBER OF ROWS OF G*/
   ERROR='D',.                         /*PRESET ERROR INDICATOR       */
   IF NN GT 0                          /*TEST SPECIFIED DIMENSIONS    */
   THEN IF MM GT 0
   THEN DO,.
        IF OPT='2'
        THEN DO,.                      /*IN CASE OF MULTIPL. S*G      */
             NN   =MM,.                /*INTERCHANGE NN AND MM        */
             MM   =N,.
             END,.
        K    =0,.
NEXTK..
        RN,CN,K=K+1,.
             DO I =1 TO NN,.           /*REPLACE H(*) BY CURRENT ROW  */
             IF OPT='2'                /*RESP. COLUMN VECTOR OF G     */
             THEN RN   =I,.
             ELSE CN   =I,.
             H(I) =G(RN,CN),.
             END,.
        LI,I =1,.
NEXTI..                                /*FOR CURRENT ROW RESP. COLUMN */
        L    =LI,.                     /*VECTOR COMPUTE I-TH ELEMENT  */
        T    =0,.
             DO J =1 TO NN,.           /*PERFORM SCALAR PRODUCT       */
             T    =T+MULTIPLY(H(J),
                  S(L),53),.
             IF J LT I
             THEN L    =L+1,.
             ELSE L    =L+J,.
             END,.
        IF OPT='2'                     /*TEST SPECIFIED MULTIPLICATION*/
        THEN RN   =I,.
        ELSE CN   =I,.
        G(RN,CN)=T,.                   /*STORE RESULTANT ELEMENT      */
        LI   =LI+I,.
        I    =I+1,.
        IF I LE NN
        THEN GO TO NEXTI,.             /*INCREMENT I                  */
        ELSE IF K LT MM
        THEN GO TO NEXTK,.             /*INCREMENT K                  */
        ERROR='0',.                    /*SUCCESSFUL OPERATION         */
        END,.
   END,.                               /*END OF PROCEDURE MMGS        */
```

Purpose:

MMGS calculates G · S if OPT='1'
S · G if OPT='2'
where G is a general and S a symmetric matrix.

Usage:

CALL MMGS (G, S, M, N, OPT);

G(M, N) - BINARY FLOAT [(53)]
Given general M by N matrix.
Resultant product matrix G · S or S · G.

S(dimension) - BINARY FLOAT [(53)]
Given symmetric N by N or M by M matrix stored in compressed form in a one-dimensional array, lower triangular part rowwise.

M - BINARY FIXED
Given row dimension of matrix A.

N - BINARY FIXED
Given column dimension of matrix A.

OPT - CHARACTER (1)
Given option for selection of operation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

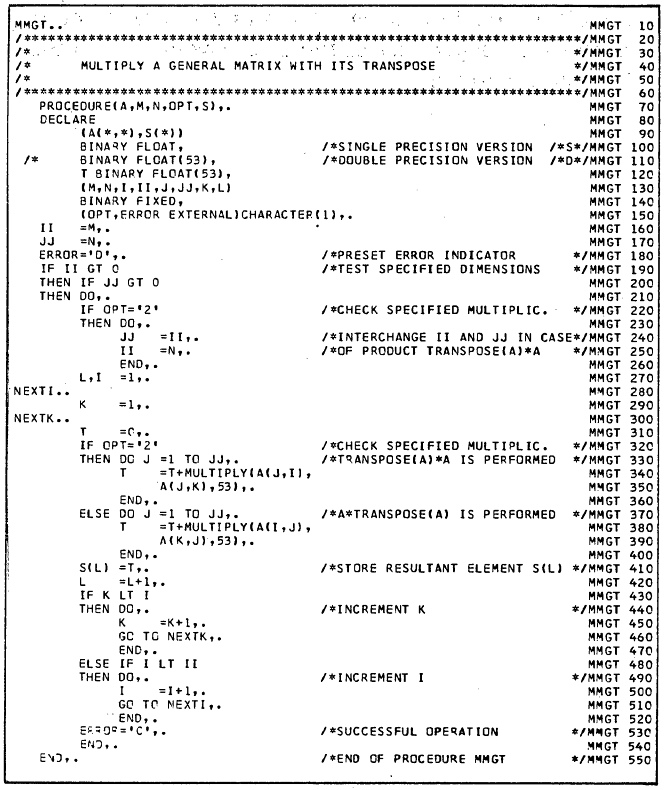
ERROR ='D' means errors in specified dimensions M, N. Any value of OPT different from '2' is treated as if it were '1'.

Scalar products are accumulated in double-precision arithmetic.

Method:

Standard multiplication is performed; the general product is generated in the storage locations occupied by G.

- Subroutine MMGT



```
MMGT..
/*********************************************************************/
/*                                                                   */
/*      MULTIPLY A GENERAL MATRIX WITH ITS TRANSPOSE                 */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(A,M,N,OPT,S),.
   DECLARE
         (A(*,*),S(*))
         BINARY FLOAT,                /*SINGLE PRECISION VERSION  /*S*/
 /*      BINARY FLOAT(53),            /*DOUBLE PRECISION VERSION  /*D*/
         T BINARY FLOAT(53),
         (M,N,I,II,J,JJ,K,L)
         BINARY FIXED,
         (OPT,ERROR EXTERNAL)CHARACTER(1),.
   II    =M,.
   JJ    =N,.
   ERROR='D',.                        /*PRESET ERROR INDICATOR       */
   IF II GT 0                         /*TEST SPECIFIED DIMENSIONS    */
   THEN IF JJ GT 0
   THEN DO,.
         IF OPT='2'                   /*CHECK SPECIFIED MULTIPLIC.   */
         THEN DO,.
              JJ    =II,.             /*INTERCHANGE II AND JJ IN CASE*/
              II    =N,.              /*OF PRODUCT TRANSPOSE(A)*A    */
              END,.
         L,I  =1,.
NEXTI..
         K    =1,.
NEXTK..
         T    =0,.
         IF OPT='2'                   /*CHECK SPECIFIED MULTIPLIC.   */
         THEN DO J =1 TO JJ,.         /*TRANSPOSE(A)*A IS PERFORMED  */
              T     =T+MULTIPLY(A(J,I),
                    A(J,K),53),.
              END,.
         ELSE DO J =1 TO JJ,.         /*A*TRANSPOSE(A) IS PERFORMED  */
              T     =T+MULTIPLY(A(I,J),
                    A(K,J),53),.
              END,.
         S(L) =T,.                    /*STORE RESULTANT ELEMENT S(L) */
         L    =L+1,.
         IF K LT I
         THEN DO,.                    /*INCREMENT K                  */
              K     =K+1,.
              GO TO NEXTK,.
              END,.
         ELSE IF I LT II
         THEN DO,.                    /*INCREMENT I                  */
              I     =I+1,.
              GO TO NEXTI,.
              END,.
         ERROR='C',.                  /*SUCCESSFUL OPERATION         */
         END,.
   END,.                              /*END OF PROCEDURE MMGT        */
```

Purpose:

MMGT calculates $A \cdot A^T$ if OPT='1'
$A^T \cdot A$ if OPT ='2'

Usage:

CALL MMGT (A, M, N, OPT, S);

A(M, N) -	BINARY FLOAT [(53)] Given M by N matrix.
M -	BINARY FIXED Given row dimension of A.
N -	BINARY FIXED Given column dimension of A.
OPT -	CHARACTER(1) Given option for selection of operation
S(dimension) -	BINARY FLOAT [(53)] Resultant symmetric product matrix, stored in compressed form in a one-dimensional array. Dimension is $M \cdot (M+1)/2$ if OPT='1' and $N \cdot (N+1)/2$ if OPT='2'.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

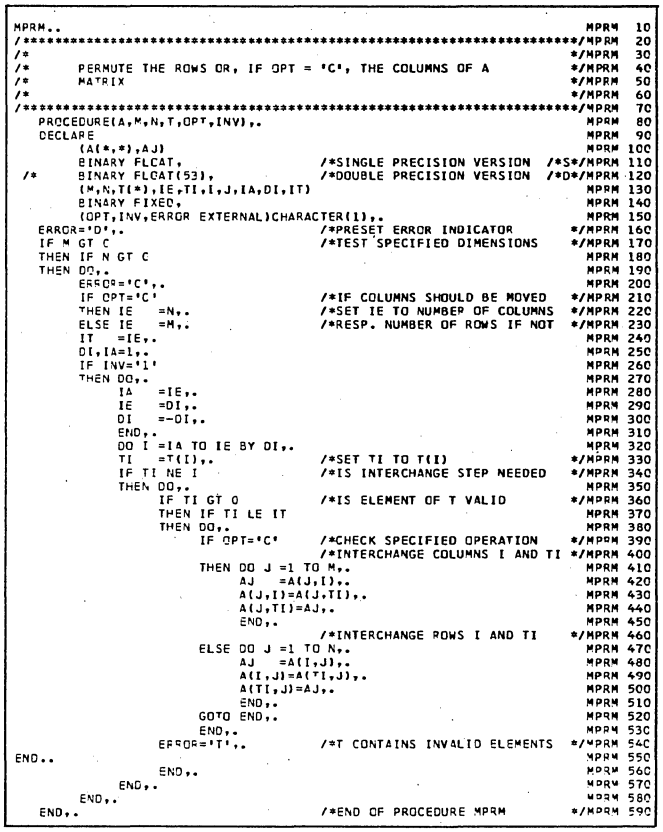
ERROR='D' means errors in specified dimensions M, N. Any value of OPT different from '2' is treated as if it were '1'.

Scalar products are accumulated in double-precision arithmetic.

Method:

Standard multiplication is performed; A · A^T is symmetric M by M, while A^T · A is symmetric N by N.

- Subroutine MPRM



```
MPRM..
/*********************************************************************/
/*                                                                   */
/*      PERMUTE THE ROWS OR, IF OPT = 'C', THE COLUMNS OF A          */
/*      MATRIX                                                       */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(A,M,N,T,OPT,INV),.
   DECLARE
        (A(*,*),AJ)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        (M,N,T(*),IE,TI,I,J,IA,DI,IT)
        BINARY FIXED,
        (OPT,INV,ERROR EXTERNAL)CHARACTER(1),.
   ERROR='D',.                         /*PRESET ERROR INDICATOR       */
   IF M GT C                           /*TEST SPECIFIED DIMENSIONS    */
   THEN IF N GT C
   THEN DO,.
        ERROR='C',.
        IF OPT='C'                     /*IF COLUMNS SHOULD BE MOVED   */
        THEN IE   =N,.                 /*SET IE TO NUMBER OF COLUMNS  */
        ELSE IE   =M,.                 /*RESP. NUMBER OF ROWS IF NOT  */
        IT   =IE,.
        DI,IA=1,.
        IF INV='1'
        THEN DO,.
             IA   =IE,.
             IE   =DI,.
             DI   =-DI,.
             END,.
             DO I =IA TO IE BY DI,.
             TI   =T(I),.              /*SET TI TO T(I)               */
             IF TI NE I                /*IS INTERCHANGE STEP NEEDED   */
             THEN DO,.
                  IF TI GT 0           /*IS ELEMENT OF T VALID        */
                  THEN IF TI LE IT
                  THEN DO,.
                       IF OPT='C'      /*CHECK SPECIFIED OPERATION    */
                                       /*INTERCHANGE COLUMNS I AND TI */
                       THEN DO J =1 TO M,.
                            AJ   =A(J,I),.
                            A(J,I)=A(J,TI),.
                            A(J,TI)=AJ,.
                            END,.
                                       /*INTERCHANGE ROWS I AND TI    */
                       ELSE DO J =1 TO N,.
                            AJ   =A(I,J),.
                            A(I,J)=A(TI,J),.
                            A(TI,J)=AJ,.
                            END,.
                       GOTO END,.
                       END,.
                  ERROR='T',.          /*T CONTAINS INVALID ELEMENTS  */
END..
                  END,.
             END,.
        END,.
   END,.                               /*END OF PROCEDURE MPRM        */
```

Purpose:

MPRM permutes rows (if OPT='R') or columns (if OPT='C') of a given matrix A according to the permutation P (if INV='0') or its inverse P^{-1} (if INV='1'). The permutation P is given in the form of its transposition vector T.

Usage:

CALL MPRM (A, M, N, T, OPT, INV);

A(M, N) - BINARY FLOAT [(53)]
Given M by N matrix.
Resultant matrix.

M - BINARY FIXED
Given number of rows of A.

N - BINARY FIXED
Given number of columns of A.

T(range) - BINARY FIXED
Given transposition vector. Its dimension range equals M if OPT='R' and N if OPT='C'.

OPT - CHARACTER(1)
Given option specifying row or column permutation.

INV - CHARACTER(1)
Given option specifying whether permutation P or inverse permutation P^{-1} is applied.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='D' means error in specified dimensions.
ERROR='T' means invalid transposition vector.

If some element t_i of T does not satisfy $1 \le t_i \le$ range (invalid transposition vector), then the value of this element is interpreted as if it were equal to i (no interchange).

Any value of OPT different from 'C' is interpreted as if it were 'R'.

Any value of INV different from '1' is interpreted as if it were '0'.

Method:

Permutation of A is performed by successively interchanging rows (if OPT='R') or columns (if OPT='C'), i and t_i for i = 1 up to range if INV='0' and for i = range down to 1 if INV ='1'.

In case $i = t_i$ no interchange takes place.

Mathematical Background:

The resultant A is calculated as the product

$$I_{m,t_m} \cdot I_{m-1,t_{m-1}} \ldots I_{1,t_1} \cdot A$$

if OPT='R', INV='0'

$$I_{1,t_1} \cdot I_{2,t_2} \ldots I_{m,t_m} \cdot A$$

if OPT='R', INV='1'

$$A \cdot I_{1,t_1} \cdot I_{2,t_2} \ldots I_{n,t_n}$$

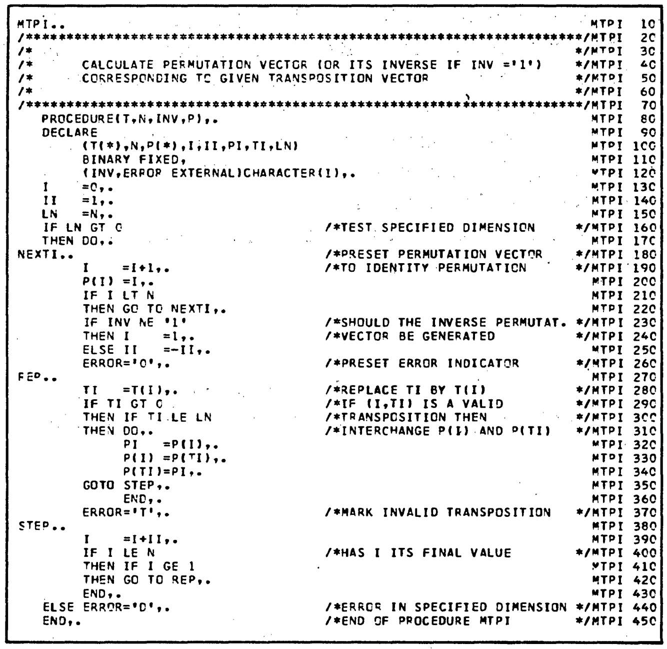
if OPT ='C', INV='0'

$$A \cdot I_{n,t_n} \cdot I_{n-1,t_{n-1}}, \ldots I_{1,t_1}$$

if OPT='C', INV='1'

For notational details see MPIT.

- <u>Subroutine</u> MTPI



```
MTPI..
/*************************************************************************/
/*                                                                       */
/*      CALCULATE PERMUTATION VECTOR (OR ITS INVERSE IF INV ='1')        */
/*      CORRESPONDING TO GIVEN TRANSPOSITION VECTOR                      */
/*                                                                       */
/*************************************************************************/
   PROCEDURE(T,N,INV,P),.
   DECLARE
        (T(*),N,P(*),I,II,PI,TI,LN)
        BINARY FIXED,
        (INV,ERROR EXTERNAL)CHARACTER(1),.
   I    =0,.
   II   =1,.
   LN   =N,.
   IF LN GT 0                              /*TEST SPECIFIED DIMENSION     */
   THEN DO,.
NEXTI..                                    /*PRESET PERMUTATION VECTOR    */
        I    =I+1,.                        /*TO IDENTITY PERMUTATION      */
        P(I) =I,.
        IF I LT N
        THEN GO TO NEXTI,.
        IF INV NE '1'                      /*SHOULD THE INVERSE PERMUTAT. */
        THEN I     =1,.                    /*VECTOR BE GENERATED          */
        ELSE II    =-II,.
        ERROR='0',.                        /*PRESET ERROR INDICATOR       */
REP..
        TI   =T(I),.                       /*REPLACE TI BY T(I)           */
        IF TI GT 0                         /*IF (I,TI) IS A VALID         */
        THEN IF TI LE LN                   /*TRANSPOSITION THEN           */
        THEN DO,.                          /*INTERCHANGE P(I) AND P(TI)   */
             PI   =P(I),.
             P(I) =P(TI),.
             P(TI)=PI,.
        GOTO STEP,.
             END,.
        ERROR='T',.                        /*MARK INVALID TRANSPOSITION   */
STEP..
        I    =I+II,.
        IF I LE N                          /*HAS I ITS FINAL VALUE        */
        THEN IF I GE 1
        THEN GO TO REP,.
        END,.
   ELSE ERROR='D',.                        /*ERROR IN SPECIFIED DIMENSION */
   END,.                                   /*END OF PROCEDURE MTPI        */
```

Purpose:

MTPI calculates the permutation vector if INV='0' and the inverse permutation vector if INV='1' from a given transposition vector.

Usage:

CALL MTPI (T, N, INV, P);

T(N) - BINARY FIXED
Given transposition vector.
N - BINARY FIXED
Given dimension of vectors T and P.
INV - CHARACTER(1)
Given option for selection of operation.
P(N) - BINARY FIXED
Resultant vector containing the permutation vector of permutation or inverse permutation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='D' means N is less than 1.
ERROR='T' means T contains elements outside the range (1, N).

A value of INV different from '1' is interpreted as if it were '0'.

Method:

Vector P is preset to the identity permutation P=(1,...,N). Interchanging successively the components i and t_i within P results in the permutation vector belonging to T if i runs from 1 up to N and to the inverse permutation if i runs backward from N down to 1.

Mathematical Background:

See MPIT for notation and definitions on permutation and transposition vectors.

The permutation vector $P=(p_1,\ldots,p_n)$ corresponding to the transposition vector $T = (t_1,\ldots,t_n)$ is defined through:

$$I[k, p_k] = I_{n,t_n} \cdot I_{n-1,t_{n-1}} \cdots I_{1,t_1} \cdot I$$

The elementary matrices I_{jk} are symmetric and orthogonal, that is,

$$I_{jk} = I_{jk}^{T} = I_{jk}^{-1}$$

Therefore, the inverse permutation vector is defined by:

$$I[k, q_k] = I_{1,t_1} \cdot I_{2,t_2} \cdots I_{n,t_n} \cdot I$$

Programming Considerations:

For valid transposition vectors it is necessary that $1 \le t_i \le n$ for all $i = 1, 2, \ldots, n$. As soon as a given transposition vector is detected nonvalid, the error indicator is set to T and further calculation is bypassed.

- Subroutine MPIT

```
MPIT..
/**********************************************************************/
/*                                                                    */
/*      CALCULATE THE INVERSE PERMUTATION VECTOR OR, IF OPT = 'T',    */
/*      THE TRANSPOSITION VECTORS OF THE GIVEN AND INVERSE            */
/*      PERMUTATIONS                                                  */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(P,N,OPT,PI),.
   DECLARE
        (P(*),N,PI(*),LN,J,P1,P2)
        BINARY FIXED,
        (OPT,ERROR EXTERNAL)CHARACTER(1),.
   LN,J =N,.
   IF LN GT 0                         /*TEST SPECIFIED DIMENSION      */
   THEN DO,.
REP..
        PI(J)=0,.                     /*PRESET RESULTING VALUES IN    */
        J    =J-1,.                   /*ORDER TO CHECK PERMUTATION    */
        IF J GT 0
        THEN GO TO REP,.
        ERROR='P',.                   /*PRESET ERROR INDICATOR        */
NEXTJ..
        J    =J+1,.
        P1   =P(J),.                  /*SET P1 TO P(J)                */
        IF P1 LE LN                   /*FEASIBILITY TEST..            */
        THEN IF P1 GT 0               /*IS 1 LE P1 LE N, AND IS       */
        THEN IF PI(P1)=0              /*P1 DIFF. FROM PREVIOUS VALUES*/
        THEN DO,.
             PI(P1)=J,.               /*SET P1-TH ELEMENT OF PI TO J */
             IF J LT LN               /*HAS J ITS FINAL VALUE         */
             THEN GO TO NEXTJ,.
             ERROR='0',.              /*VALID PERMUTATION VECTOR      */
             IF OPT='T'               /*IF SPECIFIED THEN TRANSPOS.   */
             THEN DO J =1 TO LN,.     /*VECTORS ARE CALCULATED        */
                  P1   =P(J),.
                  P2   =PI(J),.
                  P(P2)=P1,.
                  PI(P1)=P2,.
                  END,.
             END,.
        END,.
   ELSE ERROR='D',.                   /*ERROR IN SPECIFIED DIMENSION */
   END,.                              /*END OF PROCEDURE MPIT         */
```

Purpose:

MPIT calculates the permutation vector corresponding to the inverse of a given permutation if OPT='I' and the transposition vectors of the given permutation and of its inverse if OPT='T'.

Usage:

CALL MPIT (P, N, OPT, PI);

P(N) - BINARY FIXED
Given permutation vector of given permutation.
Resultant transposition vector of given permutation if OPT='T'; otherwise, unchanged.

N - BINARY FIXED
Given dimension of vectors P and PI.

OPT - CHARACTER(1)
Given option for selection of operation.

PI(N) - BINARY FIXED
Resultant permutation vector of inverse permutation if OPT='I' or transposition vector of inverse permutation if OPT='T'.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='D' means N is less than 1.

ERROR='P' means given permutation vector P is not a valid permutation vector. A value of OPT different from 'T' is treated as if it were "I'. PI cannot coincide with P in case OPT='I'.

Method:

In case OPT='I' as well as OPT='T' the first step is calculation of the inverse permutation vector PI combined with a check on the feasibility of given permutation vector P.

If OPT='T' a second step is performed which replaces the permutation vectors by the corresponding transposition vectors simultaneously.

Mathematical Background:

Elementary matrices I_{kl}

The elementary matrix I_{kl} is obtained from the identity matrix I by interchanging rows k and l. Multiplication of a matrix A on the left by an I_{kl} of compatible dimensions results in an interchange of rows k and l of A, while multiplication on the right interchanges columns k and l. An interchange of two elements is also called a transposition. Note that I_{kl} is symmetric and orthogonal:

$$I_{kl} = I_{kl}^T = I_{kl}^{-1}$$

Permutation vector

Let N* denote the set of integers $\{1, 2, \ldots, n\}$. A permutation is a one-to-one function that maps N* onto N*. It is fully described by the ordered n-tuple $(s_1, s_2, \ldots, s_n)$ called a permutation vector, where $s_i \in N^*$ is the function value corresponding to argument $i \in N^*$. Applying the permutation $(s_1, \ldots, s_n)$ on the rows of the n by n identity matrix I results in an orthogonal matrix $I[k, s_k]$. The notation indicates that the k-th row is identical with the s_k-th row of I for all $k = 1, 2, \ldots, n$.

If an n by n matrix A is multiplied on the left by $I[k, s_k]$, its rows get permuted according to the permutation vector $(s_1, s_2, \ldots, s_n)$.

Permutation of columns is similarly performed multiplying by the permutation matrix $I^T[k, s_k] = I[s_k, k]$ on the right-hand side.

Transposition vector

An n-term product $I_{n,t_n} \cdot I_{n-1,t_{n-1}} \cdots I_{1,t_1}$ corresponds uniquely to a permutation matrix $I[k, s_k]$. The ordered n-tuple $(t_1, t_2, \ldots, t_n)$, which fully describes the above transposition product, is called a transposition vector. The correspondence between permutation vectors and transposition vectors is not one to one: a given permutation vector $(s_1, s_2, \ldots, s_n)$ corresponds to several different transposition vectors if $n > 2$. A uniquely determined transposition vector is obtained under the additional restriction $t_i \geq i$.

The transposition vector comes in naturally when pivoting is used with Gaussian elimination technique. If, at the j-th elimination step, rows j and t_j must be interchanged for $j=1, \ldots, n$, then $(t_1, t_2, \ldots t_n)$ is the transposition vector of the permutation that was applied to the rows of the original matrix. This transposition vector is uniquely determined since $t_i \geq i$.

Permutation vector of the inverse permutation

The inverse P^{-1} of a permutation $P = (p_1, \ldots, p_n)$ has function value i corresponding to argument p_i. Let $Q = (q_1, \ldots, q_n)$ be the permutation vector of P^{-1}. $I[k, p_k]$ is orthogonal -- that is, $I^{-1}[k, p_k] = I^T[k, p_k]$. Therefore, $I[k, q_k] = I[p_k, k]$. Since $I[k, q_k] = I[p_k, q_{p_k}]$, it follows by comparison $q_{p_k} = k$.

Transposition vector of permutation

The calculation of the transposition vector $T = (t_1, t_2, \ldots, t_n)$ corresponding to the permutation vector $P = (p_1, p_2, \ldots, p_n)$ is based on the identity

$$I[k, p_k] \cdot I_{i,\, q_i} = I[k, p_k'] \qquad (1)$$

with $P' = (p_1', \ldots, p_n') = (p_1, \ldots, p_{i-1}, i,$

$$p_{i+1}, \ldots, p_{q_i-1}, q_i, p_{q_i+1}, \ldots, p_n)$$

Applying identity (1) successively for $i = 1, 2, \ldots, n$ leads to

$$I[k, p_k] \cdot I_{1,\, t_1} \cdot I_{2,\, t_2} \cdots I_{n,\, t_n} = I$$

or

$$I[k, p_k] = I_{n,t_n} \cdot I_{n-1,t_{n-1}} \cdots I_{2,\, t_2} \cdot$$

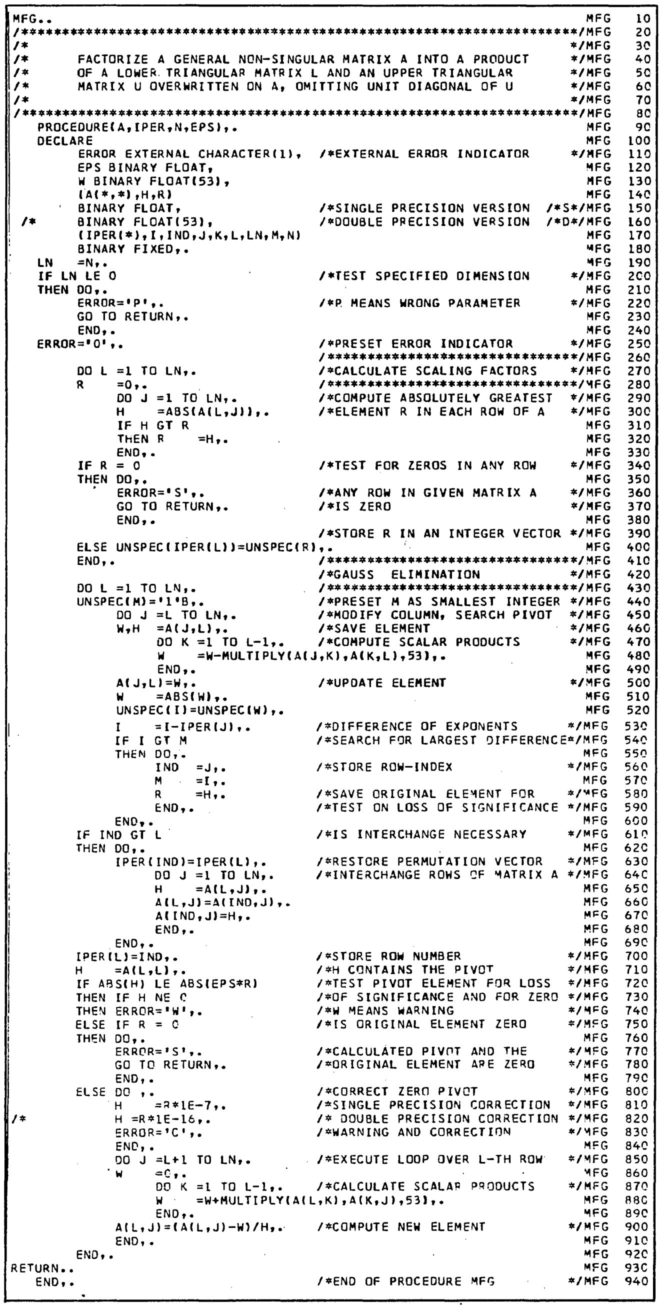
$$I_{1,t_1}$$

It is interesting to note that combining the calculation of transposition vectors of P and P^{-1} greatly improves the efficiency.

Programming Considerations:

The check on validity of the given permutation vector is performed so that all components of the vector PI are preset to zero. At the i-th step of the calculation of the inverse permutation vector, p_i is checked for $1 \le p_i \le n$, and q_{p_i} is checked for zero. If both restrictions are met q_{p_i} is reset to i. Otherwise, the error indicator is set to 'P' and further calculation is bypassed.

- <u>Subroutine</u> MFG



```
MFG..
/**********************************************************************/
/*                                                                    */
/*      FACTORIZE A GENERAL NON-SINGULAR MATRIX A INTO A PRODUCT      */
/*      OF A LOWER TRIANGULAR MATRIX L AND AN UPPER TRIANGULAR        */
/*      MATRIX U OVERWRITTEN ON A, OMITTING UNIT DIAGONAL OF U        */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,IPER,N,EPS),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */
        EPS BINARY FLOAT,
        W BINARY FLOAT(53),
        (A(*,*),H,R)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (IPER(*),I,IND,J,K,L,LN,M,N)
        BINARY FIXED,.
   LN   =N,.
   IF LN LE 0                         /*TEST SPECIFIED DIMENSION      */
   THEN DO,.
        ERROR='P',.                   /*P MEANS WRONG PARAMETER       */
        GO TO RETURN,.
        END,.
   ERROR='0',.                        /*PRESET ERROR INDICATOR        */
                                      /********************************/
        DO L =1 TO LN,.               /*CALCULATE SCALING FACTORS     */
        R    =0,.                     /********************************/
             DO J =1 TO LN,.          /*COMPUTE ABSOLUTELY GREATEST   */
             H    =ABS(A(L,J)),.      /*ELEMENT R IN EACH ROW OF A    */
             IF H GT R
             THEN R    =H,.
             END,.
        IF R = 0                      /*TEST FOR ZEROS IN ANY ROW     */
        THEN DO,.
             ERROR='S',.              /*ANY ROW IN GIVEN MATRIX A     */
             GO TO RETURN,.           /*IS ZERO                       */
             END,.
                                      /*STORE R IN AN INTEGER VECTOR */
        ELSE UNSPEC(IPER(L))=UNSPEC(R),.
        END,.                         /********************************/
                                      /*GAUSS  ELIMINATION            */
        DO L =1 TO LN,.               /********************************/
        UNSPEC(M)='1'B,.              /*PRESET M AS SMALLEST INTEGER */
             DO J =L TO LN,.          /*MODIFY COLUMN, SEARCH PIVOT   */
             W,H  =A(J,L),.           /*SAVE ELEMENT                  */
                  DO K =1 TO L-1,.    /*COMPUTE SCALAR PRODUCTS       */
                  W    =W-MULTIPLY(A(J,K),A(K,L),53),.
                  END,.
             A(J,L)=W,.               /*UPDATE ELEMENT                */
             W    =ABS(W),.
             UNSPEC(I)=UNSPEC(W),.
             I    =I-IPER(J),.        /*DIFFERENCE OF EXPONENTS       */
             IF I GT M                /*SEARCH FOR LARGEST DIFFERENCE*/
             THEN DO,.
                  IND  =J,.           /*STORE ROW-INDEX               */
                  M    =I,.
                  R    =H,.           /*SAVE ORIGINAL ELEMENT FOR     */
                  END,.               /*TEST ON LOSS OF SIGNIFICANCE */
             END,.
        IF IND GT L                   /*IS INTERCHANGE NECESSARY      */
        THEN DO,.
             IPER(IND)=IPER(L),.      /*RESTORE PERMUTATION VECTOR    */
                  DO J =1 TO LN,.     /*INTERCHANGE ROWS OF MATRIX A */
                  H    =A(L,J),.
                  A(L,J)=A(IND,J),.
                  A(IND,J)=H,.
                  END,.
             END,.
        IPER(L)=IND,.                 /*STORE ROW NUMBER              */
        H    =A(L,L),.                /*H CONTAINS THE PIVOT          */
        IF ABS(H) LE ABS(EPS*R)       /*TEST PIVOT ELEMENT FOR LOSS   */
        THEN IF H NE 0                /*OF SIGNIFICANCE AND FOR ZERO */
        THEN ERROR='W',.              /*W MEANS WARNING               */
        ELSE IF R = 0                 /*IS ORIGINAL ELEMENT ZERO      */
        THEN DO,.
             ERROR='S',.              /*CALCULATED PIVOT AND THE      */
             GO TO RETURN,.           /*ORIGINAL ELEMENT ARE ZERO     */
             END,.
        ELSE DO ,.                    /*CORRECT ZERO PIVOT            */
             H    =R*1E-7,.           /*SINGLE PRECISION CORRECTION   */
/*           H =R*1E-16,.             /* DOUBLE PRECISION CORRECTION */
             ERROR='C',.              /*WARNING AND CORRECTION        */
             END,.
             DO J =L+1 TO LN,.        /*EXECUTE LOOP OVER L-TH ROW    */
             W    =0,.
                  DO K =1 TO L-1,.    /*CALCULATE SCALAR PRODUCTS     */
                  W    =W+MULTIPLY(A(L,K),A(K,J),53),.
                  END,.
             A(L,J)=(A(L,J)-W)/H,.    /*COMPUTE NEW ELEMENT           */
             END,.
        END,.
RETURN..
   END,.                              /*END OF PROCEDURE MFG          */
```

Purpose:

MFG factorizes a general nonsingular matrix A into a product of a lower triangular matrix L and an upper triangular matrix U overwritten on A, omitting the unit diagonal of U.

Usage:

CALL MFG (A, IPER, N, EPS);

A(N, N) - BINARY FLOAT [(53)]
Given two-dimensional array.
Resultant calculated triangular factors L and U, where unit diagonal of U is not stored.

IPER(N) - BINARY FIXED
Resultant vector containing the permutations of rows of the matrix.

N - BINARY FIXED
Given order of matrix A.

EPS - BINARY FLOAT
Given relative tolerance for test on loss of significant digits.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='P' means error in specified dimension $N \leq 0$

ERROR='S' means that any row in the given matrix A is zero or that any calculated pivot and the corresponding original elements are zero; this implies that the given matrix A is singular.

ERROR='G' indicates correction. Any calculated zero pivot is modified to $R \cdot 10^{-7}$ in single precision ($R \cdot 10^{-16}$ in double precision if the corresponding original element R is nonzero).

ERROR='W' indicates a warning. A possible loss of significance may occur.

If at any factorization step the calculated pivot is equal to zero, the corresponding original element R is tested for zero. The given matrix A is interpreted as being singular if R is zero. MFG sets error indicator ERROR to 'S' and further calculation is bypassed. If R is not zero, pivot is corrected to $R \cdot 10^{-7}$ (in double precision $R \cdot 10^{-16}$) and ERROR is set to 'G'.

Method:

Calculation of the triangular factors L and U is done using the standard Gaussian elimination technique. Columnwise pivoting is built in, combined with scaling of rows (equilibration). The upper triangular matrix U is normalized so that the diagonal contains all ones, which are not stored. The given matrix A is overwritten by the resulting triangular factors L and U, omitting the unit diagonal of U.

For reference, see:

H. J. Bowdler, R. S. Martin, G. Peters, J. H. Wilkinson, "Solution of Real and Complex Systems of Linear Equations", Numerische Mathematik, Vol. 8, 1966, pp. 217-234.
A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, Vol. 2, 1967, pp. 69-71.

Mathematical Background:

Let A be a nonsingular real matrix of order n. In general, it can be factorized into a product

$$A = L \cdot U$$

where L and U are lower and upper triangular matrices respectively; U is chosen so that it has a unit diagonal.

The elements l_{ik} and u_{ik} of the factor matrices L and U are computed using the following recursive formulas:

$$l_{ik} = a_{ik} - \sum_{m=1}^{k-1} l_{im} \cdot u_{mk}$$

$$\left\{ \begin{matrix} i = 1, 2, \ldots, N \\ k = 1, 2, \ldots, i \end{matrix} \right\}$$

$$u_{ik} = \frac{1}{l_{ii}} \left(a_{ik} - \sum_{m=1}^{i-1} l_{im} \cdot u_{mk} \right)$$

$$\left\{ \begin{matrix} i = 1, 2, \ldots, N-1 \\ k = i+1, \ldots, N \end{matrix} \right\}$$

Programming Considerations:

Even if the given matrix A is nonsingular and well conditioned, the process can fail when a leading principal submatrix of A is singular; furthermore, the process is numerically unstable whenever a leading principal submatrix is ill conditioned.

In order to avoid these inconveniences, a technique of partial pivoting with an equilibration of the matrix has been introduced in MFG. Initially, the element with greatest absolute value -- say,

W_i (i=1, 2, ..., N), of each row of A is computed. The scaling factors W_i are used as weights for pivoting.

The p-th factorization step is as follows:

1. Computation of the p-th column of L:

$$l_{ip} = a_{ip} - \sum_{m=1}^{p-1} l_{im} \cdot u_{mp}$$

and overwrite l_{ip} on a_{ip} (i = p, p+1, ..., N)

2. Equilibrated partial pivoting:

Choose k so that

$$\frac{|l_{kp}|}{W_k} = \underset{i \geq p}{\text{MAX}} \left\{ \frac{|l_{ip}|}{W_i} \right\}$$

Store the integer k in the vector IPER_p and, if k > p, interchange the k-th and p-th rows. Then l_{pp} is the next pivot.

3. Computation of the p-th row of U:

$$u_{pi} = \frac{1}{l_{pp}} (a_{pi} - \sum_{m=1}^{p-1} l_{pm} \cdot u_{mi})$$

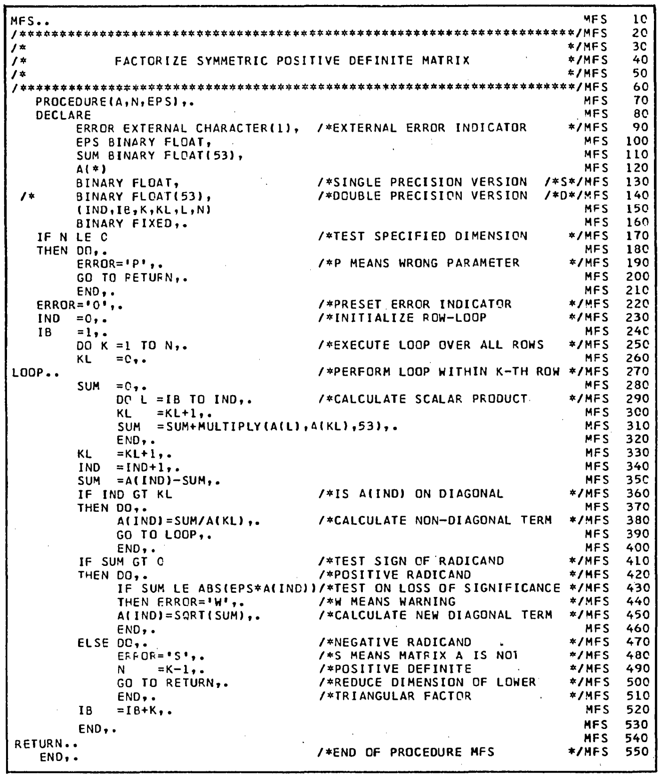
and overwrite u_{pi} on a_{pi} (i = p+1, p+2, ..., N)

The diagonal terms of U, which are 1, are not stored. For economy of storage, the scaling weights W_i are initially stored in the vector IPER. This is done using the PL/I function UNSPEC, which stores W_i in internal coded representation. This allows substituting|subtractions for divisions in the choice of pivots.

If at factorization step p the pivot l_{pp} becomes zero, the corresponding original element a_{pp} is tested for zero. The given matrix A is interpreted as being singular if a_{pp} is also zero. MFG sets error indicator ERROR to 'S' and further calculation is bypassed. In other cases zero pivot is modified to:

$$l_{pp} = a_{pp} * \left\{ \begin{array}{l} 10^{-7} \text{ in the single precision version} \\ \\ 10^{-16} \text{ in the double precision version} \end{array} \right\}$$

- Subroutine MFS



```
MFS..
/********************************************************************/
/*                                                                  */
/*          FACTORIZE SYMMETRIC POSITIVE DEFINITE MATRIX            */
/*                                                                  */
/********************************************************************/
   PROCEDURE(A,N,EPS),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR       */
        EPS BINARY FLOAT,
        SUM BINARY FLOAT(53),
        A(*)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION    /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION    /*D*/
        (IND,IB,K,KL,L,N)
        BINARY FIXED,.
   IF N LE C                          /*TEST SPECIFIED DIMENSION     */
   THEN DO,.
        ERROR='P',.                   /*P MEANS WRONG PARAMETER      */
        GO TO FETURN,.
        END,.
   ERROR='0',.                        /*PRESET ERROR INDICATOR       */
   IND  =0,.                          /*INITIALIZE ROW-LOOP          */
   IB   =1,.
        DO K =1 TO N,.                /*EXECUTE LOOP OVER ALL ROWS   */
        KL   =C,.
LOOP..                                /*PERFORM LOOP WITHIN K-TH ROW */
        SUM  =C,.
             DC L =IB TO IND,.        /*CALCULATE SCALAR PRODUCT     */
             KL   =KL+1,.
             SUM  =SUM+MULTIPLY(A(L),A(KL),53),.
             END,.
        KL   =KL+1,.
        IND  =IND+1,.
        SUM  =A(IND)-SUM,.
        IF IND GT KL                  /*IS A(IND) ON DIAGONAL        */
        THEN DO,.
             A(IND)=SUM/A(KL),.       /*CALCULATE NON-DIAGONAL TERM  */
             GO TO LOOP,.
             END,.
        IF SUM GT C                   /*TEST SIGN OF RADICAND        */
        THEN DO,.                     /*POSITIVE RADICAND            */
             IF SUM LE ABS(EPS*A(IND))/*TEST ON LOSS OF SIGNIFICANCE */
             THEN ERROR='W',.         /*W MEANS WARNING              */
             A(IND)=SQRT(SUM),.       /*CALCULATE NEW DIAGONAL TERM  */
             END,.
        ELSE DC,.                     /*NEGATIVE RADICAND            */
             ERFOR='S',.              /*S MEANS MATRIX A IS NO1      */
             N    =K-1,.              /*POSITIVE DEFINITE            */
             GO TO RETURN,.           /*REDUCE DIMENSION OF LOWER    */
             END,.                    /*TRIANGULAR FACTOR            */
        IB   =IB+K,.
        END,.
RETURN..
   END,.                              /*END OF PROCEDURE MFS         */
```

Purpose:

MFS computes a triangular factorization of a symmetric positive definite matrix using the square root method of Cholesky.

Usage:

CALL MFS (A, N, EPS);

A(N*(N+1)/2) - BINARY FLOAT [(53)]
Given one-dimensional array containing the matrix A stored rowwise in compressed form.
Resultant calculated lower triangular factor T stored rowwise in compressed form.

N - BINARY FIXED
Given order of matrix A.
Resultant order of the triangular factor T.

EPS - BINARY FLOAT
Given relative tolerance for test on loss of significant digits.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR= 'P'	means error in specified dimension: $N \leq 0$
ERROR='S'	means given matrix A is not positive definite, possibly because of severe loss of significance.
ERROR='W'	is a warning. A possible loss of significance could occur.

The lower part of the given symmetric matrix, A, is assumed to be stored in compressed form -- that is, rowwise in N*(N+ 1)/2 successive storage locations. On return the lower triangular factor T is stored in the same way.

Method:

Factorization is done using the square root method of Cholesky, which generates a lower triangular factor matrix T such that

$$T \cdot \text{transpose}\ (T) = A$$

The given matrix, A, is replaced in core by the resultant matrix, T.

For reference, see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.
A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, Vol. 2, 1967, pp. 71-72.

Mathematical Background:

The elements t_{ij} of the lower triangular matrix T are computed using the following recursive formulas:

$$t_{kk} = \sqrt{a_{kk} - \sum_{m=1}^{k-1} t_{km}^2}$$

$$t_{ik} = \frac{a_{ik} - \sum_{m=1}^{k-1} t_{im}\, t_{km}}{t_{kk}} ,$$

$i = k+1, \ldots, N, \; k=1, \ldots, N$

($\sum_{m=1}^{j}$ is to be interpreted as zero when $j < 1$.)

The determinant of A may be computed by the formula:

$$\det(A) = \prod_{k=1}^{N} t_{kk}^2$$

Programming Considerations:

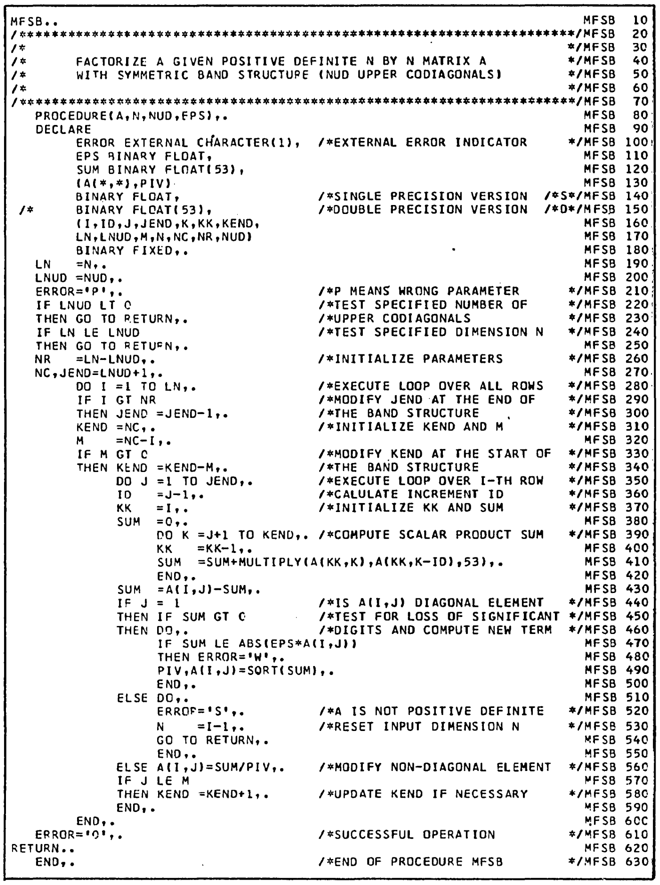
The given symmetric matrix A is assumed to be stored in compressed form. The resultant lower triangular factor T is returned in the locations of A.

If at factorization step k (k=1, 2, ..., N) the radicand is not positive, the error parameter ERROR is set to 'S', N to k-1, and further calculation is bypassed.

The error parameter ERROR is set to 'W' if any calculated radicand $\bar{r} = r - \text{SUM}$ is not greater than $| \text{EPS} \cdot r |$, where r is the original diagonal term and SUM a scalar product sum.

It should be noted that Cholesky factorization is done without pivoting.

- Subroutine MFSB



```
MFSB..
/**********************************************************************/
/*                                                                    */
/*      FACTORIZE A GIVEN POSITIVE DEFINITE N BY N MATRIX A           */
/*      WITH SYMMETRIC BAND STRUCTUPE (NUD UPPER CODIAGONALS)         */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,N,NUD,EPS),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),   /*EXTERNAL ERROR INDICATOR     */
        EPS BINARY FLOAT,
        SUM BINARY FLOAT(53),
        (A(*,*),PIV)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        (I,ID,J,JEND,K,KK,KEND,
        LN,LNUD,M,N,NC,NR,NUD)
        BINARY FIXED,.
   LN   =N,.
   LNUD =NUD,.
   ERROR='P',.                         /*P MEANS WRONG PARAMETER      */
   IF LNUD LT 0                        /*TEST SPECIFIED NUMBER OF     */
   THEN GO TO RETURN,.                 /*UPPER CODIAGONALS            */
   IF LN LE LNUD                       /*TEST SPECIFIED DIMENSION N   */
   THEN GO TO RETURN,.
   NR   =LN-LNUD,.                     /*INITIALIZE PARAMETERS        */
   NC,JEND=LNUD+1,.
        DO I =1 TO LN,.                /*EXECUTE LOOP OVER ALL ROWS   */
        IF I GT NR                     /*MODIFY JEND AT THE END OF    */
        THEN JEND =JEND-1,.            /*THE BAND STRUCTURE           */
        KEND =NC,.                     /*INITIALIZE KEND AND M        */
        M    =NC-I,.
        IF M GT 0                      /*MODIFY KEND AT THE START OF  */
        THEN KEND =KEND-M,.            /*THE BAND STRUCTURE           */
             DO J =1 TO JEND,.         /*EXECUTE LOOP OVER I-TH ROW   */
             ID   =J-1,.               /*CALULATE INCREMENT ID        */
             KK   =I,.                 /*INITIALIZE KK AND SUM        */
             SUM  =0,.
                  DO K =J+1 TO KEND,.  /*COMPUTE SCALAR PRODUCT SUM   */
                  KK   =KK-1,.
                  SUM  =SUM+MULTIPLY(A(KK,K),A(KK,K-ID),53),.
                  END,.
             SUM  =A(I,J)-SUM,.
             IF J = 1                  /*IS A(I,J) DIAGONAL ELEMENT   */
             THEN IF SUM GT 0          /*TEST FOR LOSS OF SIGNIFICANT */
             THEN DO,.                 /*DIGITS AND COMPUTE NEW TERM  */
                  IF SUM LE ABS(EPS*A(I,J))
                  THEN ERROR='W',.
                  PIV,A(I,J)=SORT(SUM),.
                  END,.
             ELSE DO,.
                  ERROR='S',.          /*A IS NOT POSITIVE DEFINITE   */
                  N    =I-1,.          /*RESET INPUT DIMENSION N      */
                  GO TO RETURN,.
                  END,.
             ELSE A(I,J)=SUM/PIV,.     /*MODIFY NON-DIAGONAL ELEMENT  */
             IF J LE M
             THEN KEND =KEND+1,.       /*UPDATE KEND IF NECESSARY     */
             END,.
        END,.
   ERROR='0',.                         /*SUCCESSFUL OPERATION         */
RETURN..
   END,.                               /*END OF PROCEDURE MFSB        */
```

Purpose:

MFSB computes a triangular factorization of a symmetric positive definite band matrix using the square root method of Cholesky.

Usage:

CALL MFSB (A, N, NUD, EPS);

A(N, NUD+1) - BINARY FLOAT [(53)]
Given two-dimensional array containing the upper part of a symmetric band matrix A with NUD upper codiagonals.
Each row starts with its diagonal element.
Resultant calculated upper band factor T.

N - BINARY FIXED
Given number of rows of matrix A.
Resultant number of rows of upper band factor T.

NUD - BINARY FIXED
Given number of upper codiagonals of A.

EPS - BINARY FLOAT
Given relative tolerance for test on loss of significant digits.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='P' - means error in specified dimensions:
NUD < 0 or N ≤ NUD

ERROR='S' - means any calculated pivot is not positive -- that is, given matrix A is not positive definite. This is possibly due to a severe loss of significance.

ERROR='W' - is a warning indicating possible loss of significance.

The upper part of symmetric band matrix A, consisting of the main diagonal and NUD upper codiagonals, is assumed to be stored rowwise in array A(N, NUD+1) starting with its diagonal elements. Thus, A(i, 1) are the diagonal elements of the given band matrix A (i = 1, 2, ..., N). On return, the upper band factor T is stored in the same way in the locations of A.

Input parameters N and NUD should satisfy the following restrictions:

$0 \le NUD < N$

Method:

Factorization is done using the square root method of Cholesky. This generates the upper band factor T such that

T · transpose (T) = A

The given A is replaced by the resultant T.

For reference see:

H. Rutishauser, "Algorithmus 1 - Lineares Gleichungssystem mit symmetrischer positiv-definiter Bandmatrix nach Cholesky", Computing (Archives for Electronic Computing), Vol. 1, iss. 1, 1966, pp. 77-78.

Mathematical Background:

For the elements a_{ik} of a symmetric band matrix with NUD upper codiagonals, the following is true:

$$a_{ik} = 0 \quad \text{if} \quad |i - k| > \text{NUD}$$

The elements t_{ik} of the upper factorized matrix T are computed using the following recursive formula:

$$t_{ik} = \frac{1}{t_{ii}} \left[a_{ik} - \sum_{m=m_0}^{i-1} t_{mi} \cdot t_{mk} \right]$$

$$m_0 = \max(1, k\text{-NUD}) \quad \begin{array}{l} i=1, 2, \ldots, N \\ k=i+1, \ldots, \\ \min(i + \text{NUD}, N) \end{array}$$

(any symbol $\sum_{m=m_0}^{r} X_m$ is to be interpreted as zero if $r < m_0$)

In the special case $i = k$ (diagonal elements), the above equation may be written:

$$t_{kk} = \sqrt{a_{kk} - \sum_{m=m_0}^{k-1} t_{mk}^2} \; ;$$

$$k = 1, 2, \ldots, N \quad m_0 = \max(1, k\text{-NUD})$$

The resultant upper factor T has band structure again, because the following is true:

$$t_{ik} = 0 \quad \text{if} \quad k > i + \text{NUD}$$

Programming Considerations:

The upper part of the symmetric positive definite band matrix A, consisting of the main diagonal and NUD upper codiagonals, is assumed to be stored rowwise in the two-dimensional array A(N, NUD+ 1) such that A(i, 1) are the diagonal elements (i=1, 2, ..., N). Therefore, the elements A(i, k) of array A with i+k > N are irrelevant; they are not touched within MFSB. The resultant upper band factor T is returned in the locations of A.

If, at factorization step m(m=1, 2, ..., N), the radicand is not positive, error parameter ERROR is set to 'S', dimension N to m - 1, and further calculation is bypassed.

The error character is set to 'W' if any calculated radicand $\bar{r} = r -$ SUM is positive but no longer greater than $|\text{EPS} \cdot r|$, where r means the original diagonal term and SUM a scalar product sum.

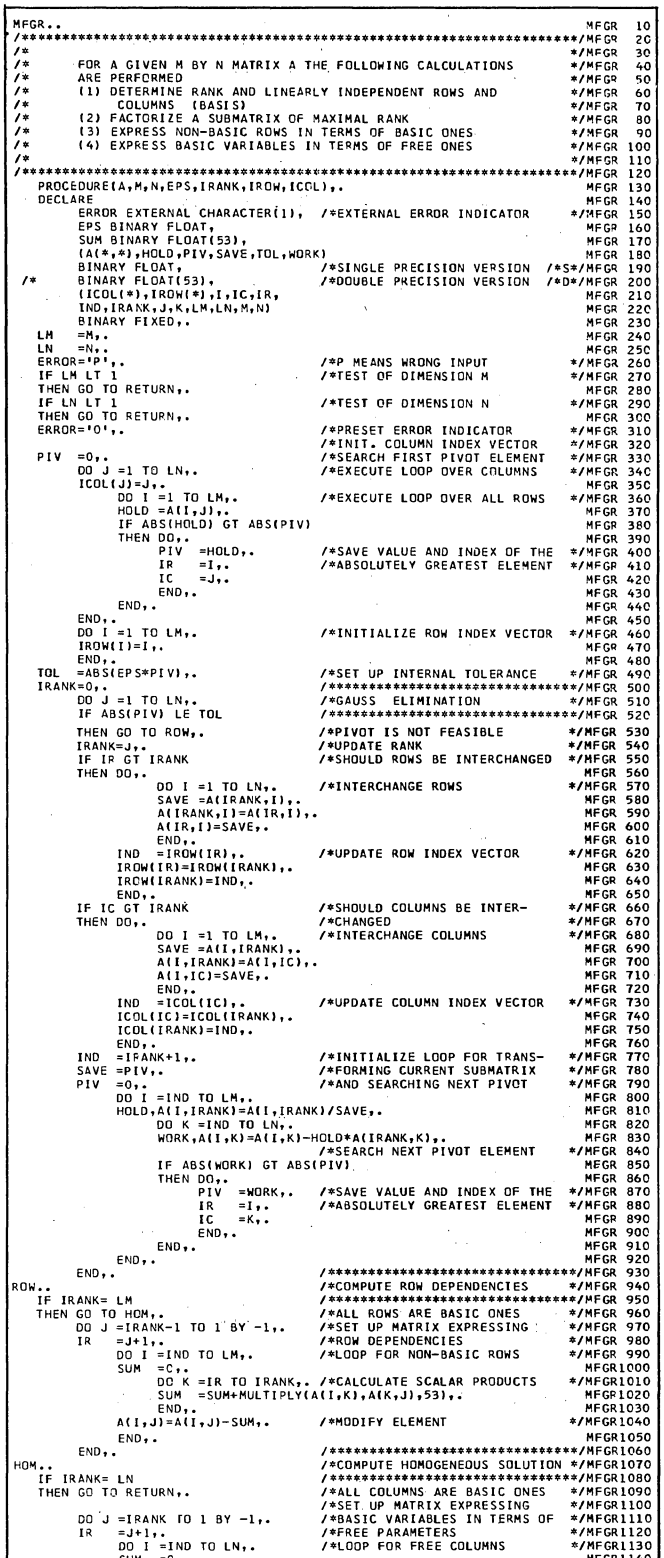
The input parameters N and NUD must satisfy the restriction:

$$0 \leq \text{NUD} < N$$

Otherwise, ERROR is set to 'P'.

It should be noted that Cholesky's factorization is done without pivoting.

• Subroutine MFGR



```
MFGR..
/**********************************************************************/
/*                                                                    */
/*      FOR A GIVEN M BY N MATRIX A THE FOLLOWING CALCULATIONS        */
/*      ARE PERFORMED                                                 */
/*      (1) DETERMINE RANK AND LINEARLY INDEPENDENT ROWS AND          */
/*           COLUMNS  (BASIS)                                         */
/*      (2) FACTORIZE A SUBMATRIX OF MAXIMAL RANK                     */
/*      (3) EXPRESS NON-BASIC ROWS IN TERMS OF BASIC ONES             */
/*      (4) EXPRESS BASIC VARIABLES IN TERMS OF FREE ONES             */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,M,N,EPS,IRANK,IROW,ICOL),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),   /*EXTERNAL ERROR INDICATOR     */
        EPS BINARY FLOAT,
        SUM BINARY FLOAT(53),
        (A(*,*),HOLD,PIV,SAVE,TOL,WORK)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        (ICOL(*),IROW(*),I,IC,IR,
        IND,IRANK,J,K,LM,LN,M,N)
        BINARY FIXED,.
   LM   =M,.
   LN   =N,.
   ERROR='P',.                         /*P MEANS WRONG INPUT          */
   IF LM LT 1                          /*TEST OF DIMENSION M          */
   THEN GO TO RETURN,.
   IF LN LT 1                          /*TEST OF DIMENSION N          */
   THEN GO TO RETURN,.
   ERROR='0',.                         /*PRESET ERROR INDICATOR       */
                                       /*INIT. COLUMN INDEX VECTOR    */
   PIV  =0,.                           /*SEARCH FIRST PIVOT ELEMENT   */
        DO J =1 TO LN,.                /*EXECUTE LOOP OVER COLUMNS    */
        ICOL(J)=J,.
             DO I =1 TO LM,.           /*EXECUTE LOOP OVER ALL ROWS   */
             HOLD =A(I,J),.
             IF ABS(HOLD) GT ABS(PIV)
             THEN DO,.
                  PIV  =HOLD,.         /*SAVE VALUE AND INDEX OF THE  */
                  IR   =I,.            /*ABSOLUTELY GREATEST ELEMENT  */
                  IC   =J,.
                  END,.
             END,.
        END,.
        DO I =1 TO LM,.                /*INITIALIZE ROW INDEX VECTOR  */
        IROW(I)=I,.
        END,.
   TOL  =ABS(EPS*PIV),.                /*SET UP INTERNAL TOLERANCE    */
   IRANK=0,.                           /*******************************/
        DO J =1 TO LN,.                /*GAUSS  ELIMINATION           */
        IF ABS(PIV) LE TOL             /*******************************/
        THEN GO TO ROW,.               /*PIVOT IS NOT FEASIBLE        */
        IRANK=J,.                      /*UPDATE RANK                  */
        IF IR GT IRANK                 /*SHOULD ROWS BE INTERCHANGED  */
        THEN DO,.
                  DO I =1 TO LN,.      /*INTERCHANGE ROWS             */
                  SAVE =A(IRANK,I),.
                  A(IRANK,I)=A(IR,I),.
                  A(IR,I)=SAVE,.
                  END,.
             IND  =IROW(IR),.          /*UPDATE ROW INDEX VECTOR      */
             IROW(IR)=IROW(IRANK),.
             IROW(IRANK)=IND,.
             END,.
        IF IC GT IRANK                 /*SHOULD COLUMNS BE INTER-     */
        THEN DO,.                      /*CHANGED                      */
                  DO I =1 TO LM,.      /*INTERCHANGE COLUMNS          */
                  SAVE =A(I,IRANK),.
                  A(I,IRANK)=A(I,IC),.
                  A(I,IC)=SAVE,.
                  END,.
             IND  =ICOL(IC),.          /*UPDATE COLUMN INDEX VECTOR   */
             ICOL(IC)=ICOL(IRANK),.
             ICOL(IRANK)=IND,.
             END,.
        IND  =IRANK+1,.                /*INITIALIZE LOOP FOR TRANS-   */
        SAVE =PIV,.                    /*FORMING CURRENT SUBMATRIX    */
        PIV  =0,.                      /*AND SEARCHING NEXT PIVOT     */
             DO I =IND TO LM,.
             HOLD,A(I,IRANK)=A(I,IRANK)/SAVE,.
                  DO K =IND TO LN,.
                  WORK,A(I,K)=A(I,K)-HOLD*A(IRANK,K),.
                                       /*SEARCH NEXT PIVOT ELEMENT    */
                  IF ABS(WORK) GT ABS(PIV)
                  THEN DO,.
                       PIV  =WORK,.    /*SAVE VALUE AND INDEX OF THE  */
                       IR   =I,.       /*ABSOLUTELY GREATEST ELEMENT  */
                       IC   =K,.
                       END,.
                  END,.
             END,.
        END,.                          /*******************************/
ROW..                                  /*COMPUTE ROW DEPENDENCIES     */
   IF IRANK= LM                        /*******************************/
   THEN GO TO HOM,.                    /*ALL ROWS ARE BASIC ONES      */
        DO J =IRANK-1 TO 1 BY -1,.     /*SET UP MATRIX EXPRESSING     */
        IR   =J+1,.                    /*ROW DEPENDENCIES             */
             DO I =IND TO LM,.         /*LOOP FOR NON-BASIC ROWS      */
             SUM  =C,.
                  DO K =IR TO IRANK,. /*CALCULATE SCALAR PRODUCTS    */
                  SUM  =SUM+MULTIPLY(A(I,K),A(K,J),53),.
                  END,.
             A(I,J)=A(I,J)-SUM,.       /*MODIFY ELEMENT               */
             END,.
        END,.                          /*******************************/
HOM..                                  /*COMPUTE HOMOGENEOUS SOLUTION */
   IF IRANK= LN                        /*******************************/
   THEN GO TO RETURN,.                 /*ALL COLUMNS ARE BASIC ONES   */
                                       /*SET UP MATRIX EXPRESSING     */
        DO J =IRANK TO 1 BY -1,.       /*BASIC VARIABLES IN TERMS OF  */
        IR   =J+1,.                    /*FREE PARAMETERS              */
             DO I =IND TO LN,.         /*LOOP FOR FREE COLUMNS        */
             SUM  =0,.
                  DO K =IR TO IRANK,. /*CALCULATE SCALAR PRODUCTS    */
                  SUM  =SUM+MULTIPLY(A(J,K),A(K,I),53),.
                  END,.
             A(J,I)=-(A(J,I)+ SUM)/A(J,J),.
             END,.
        END,.
RETURN..
   END,.                               /*END OF PROCEDURE MFGR        */
```

Purpose:

For a given general rectangular matrix, MFGR performs the following:

1. Determines rank and linearly independent rows and columns (basis)
2. Factorizes a submatrix of maximal rank
3. Expresses nonbasic rows in terms of basic rows
4. Expresses basic variables in terms of free variables

Usage:

CALL MFGR (A, M, N, EPS, IRANK, IROW, ICOL);

A(M, N) - BINARY FLOAT [(53)]
Given general matrix with M rows and N columns.
Resultant calculated triangular factors L, U and submatrices C, H, D.

M - BINARY FIXED
Given number of rows of matrix A.

N - BINARY FIXED
Given number of columns of matrix A.

EPS - BINARY FLOAT
Given relative tolerance for test on zero.

IRANK - BINARY FIXED
Resultant rank of given matrix.

IROW(M) - BINARY FIXED
Resultant vector containing the subscripts of basic rows in IROW(1) up to IROW(IRANK).

ICOL(N) - BINARY FIXED
Resultant vector containing the subscripts of basic columns in ICOL(1) up to ICOL(IRANK).

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='P' means error in specified dimensions: $M \leq 0$ and/or $N \leq 0$

Calculation of the rank of given matrix A is most critical. It is not claimed that MFGR will give the correct rank in all cases, because of the intrinsic difficulty caused by performing calculations with a finite number of digits.

Suggested range for values of EPS is $(10^{-4}, 10^{-6})$ in single precision and $(10^{-8}, 10^{-15})$ in double precision.

Method:

Calculation of the rank IRANK and of the triangular factors L and U is done using the standard Gaussian elimination technique with complete pivoting. The lower triangular matrix L is normalized so that the diagonal contains all ones, which are not stored. The subdiagonal part of L and the upper triangular factor U are stored in the locations of the given matrix A.

In case A is singular, the triangular factors L and U only of a submatrix of maximal rank are retained. The remaining parts of the resultant matrix give the dependencies of rows and columns and the solution of the homogeneous matrix equation $A \cdot X = 0$.

For reference see:

A. S. Householder, The Theory of Matrices in Numerical Analysis, 1965, pp. 125-130.

Mathematical Background:

Interchange information

Gauss elimination with complete pivoting implies that the rows and columns of the given M by N matrix A are interchanged at each elimination step if necessary. The interchange information is recorded in two integer vectors IROW and ICOL:

The i-th $\left\{ \begin{matrix} \text{row} \\ \text{column} \end{matrix} \right\}$ of the interchanged matrix corresponds

to the $\left\{ \begin{matrix} \text{IROW(i)-th row} \\ \text{ICOL(i)-th column} \end{matrix} \right\}$ in the original matrix, where initially

$$\text{IROW(i)=i and ICOL(i)=i for } i = \left\{ \begin{matrix} 1, 2, \ldots, M \\ 1, 2, \ldots, N \end{matrix} \right\}$$

At the i-th elimination step the interchanged matrix is denoted by A^i.

First elimination step

After pivoting, the interchanged matrix A^1 is uniquely expressed as:

$$A^1 = L^1 \cdot D^1 \cdot U^1$$

by imposing the following conditions:

1. U^1 is the N by N identity matrix except for the first row.
2. L^1 is the M by M identity matrix except for the first column. The first diagonal element has a value of one.
3. D^1 is an M by N matrix with first diagonal element equal to one, while all remaining elements of the first row and column are equal to zero.

Partitioning of matrices A^1, L^1, D^1, U^1 leads to:

$$\begin{pmatrix} a^1_{11} & A^1_{12} \\ A^1_{21} & A^1_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ L^1_{21} & I \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & D^1_{22} \end{pmatrix} \cdot \begin{pmatrix} u^1_{11} & U^1_{12} \\ 0 & I \end{pmatrix}$$

where:

$$a^1_{11} = u^1_{11}$$

$$A^1_{12} = U^1_{12}$$

$$A^1_{21} = L^1_{21} \cdot u^1_{11}$$

$$A^1_{22} = L^1_{21} \cdot U^1_{12} + D^1_{22}$$

This implies the following:

1. The elements of the first column of U^1 are

$$u^1_{1k} = a^1_{1k} \quad (k = 1, 2, 3, \ldots, N)$$

2. The elements of the first column of L^1 are

$$l^1_{11} = 1 \,;\, l^1_{i1} = \frac{a^1_{i1}}{a^1_{11}} \quad (i = 2, 3, \ldots, M)$$

3. The elements of submatrix D^1_{22} of D^1 are

$$d^1_{ik} = a^1_{ik} - l^1_{i1} \cdot u^1_{1k} = a^1_{ik} - \frac{a^1_{i1} \cdot a^1_{ik}}{a^1_{11}}$$

$$\begin{aligned} i &= 2,\ 3,\ \ldots,\ M \\ k &= 2,\ 3,\ \ldots,\ N \end{aligned}$$

Note that it is possible to record all nontrivial information about L^1, D^1, U^1 in the storage locations originally occupied by A, storing only:

$$\begin{pmatrix} u^1_{11} & U^1_{12} \\ L^1_{21} & D^1_{22} \end{pmatrix}$$

Second elimination step

Assume D^1_{22} is not zero in the sense that all its elements are absolutely greater than an internal tolerance TOL. The complete pivoting in D^1_{22} implies that matrix A^1 possibly is interchanged, giving A^2:

$$A^2 = \begin{pmatrix} 1 & 0 \\ L^2_{21} & I \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & D^2_{22} \end{pmatrix} \cdot \begin{pmatrix} u^1_{11} & U^2_{12} \\ 0 & I \end{pmatrix}$$

Now D^2_{22} may be expressed uniquely in the form:

$$D^2_{22} = \begin{pmatrix} 1 & 0 \\ L^2_{32} & I \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & D^2_{33} \end{pmatrix} \cdot \begin{pmatrix} u^2_{22} & U^2_{23} \\ 0 & I \end{pmatrix}$$

It is easily seen that

$$A^2 = L^2 \cdot D^2 \cdot U^2$$

where

$$L^2 = \begin{pmatrix} 1 & 0 & 0 \\ l^2_{21} & 1 & 0 \\ L^2_{31} & L^2_{32} & I \end{pmatrix}$$

$$D^2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & D^2_{33} \end{pmatrix}$$

$$U^2 = \begin{pmatrix} u^1_{11} & u^2_{12} & U^2_{13} \\ 0 & u^2_{22} & U^2_{23} \\ 0 & 0 & I \end{pmatrix}$$

Final elimination step

At the next step D^2_{33} is factorized, and so on. Now assume that $D^r_{r+1,\ r+1}$ equals zero -- that is, that all its elements are absolutely less than or equal to TOL. This is interpreted as matrix A has the rank r and the result is the factorization:

$$A^r = L^r \cdot D^r \cdot U^r$$

Neglecting the small elements in $D^r_{r+1,\ r+1}$ this may be written as:

$$A^r = \begin{pmatrix} L \\ LR \end{pmatrix} \cdot (U,\ UR)$$

with

$$L = \begin{pmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 \\ l^2_{21} & 1 & & & & 0 \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ l^r_{r1} & l^r_{r2} & \cdot & \cdot & \cdot & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} u^1_{11} & u^2_{12} & \cdot & \cdot & \cdot & u^r_{1r} \\ 0 & u^2_{22} & \cdot & & & u^r_{2r} \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & u^r_{rr} \end{pmatrix}$$

$$LR = (L^r_{r+1,\ 1},\ L^r_{r+1,\ 2},\ \ldots,\ L^r_{r+1,\ r})$$

$$UR = \begin{pmatrix} U^r_{1,\ r+1} \\ U^r_{2,\ r+1} \\ \cdot \\ \cdot \\ \cdot \\ U^r_{r,\ r+1} \end{pmatrix}$$

L is a lower triangular matrix of order r with unit diagonal.

U is an r by r upper triangular matrix.

LR is an (M-r) by r matrix; if the given matrix A is row regular (that is, r=M), LR is absent in the final factorization.

UR is an r by (N-r) matrix; if the given matrix A is column regular (that is, r = N), UR is absent in the final factorization.

Further calculations

The problem of matrix factorization arises in connection with the solution of systems of equations $A \cdot X = R$. Three different cases must be distinguished:

1. $r = M = N$

   A is nonsingular, and $A \cdot X = R$ has a unique solution.

2. $r < M$

   A is not row regular; solutions of $A \cdot X = R$ exist only if the linear combinations among the rows of A are also valid among the rows of R.

3. $r < N$

   A is not column regular; $A \cdot X = 0$ has nontrivial solutions.

The cases (2) and (3) may occur together. The solution, if it exists, is uniquely determined for r=N; otherwise, it contains N-r free parameters. It is quite natural to ask for the linear combinations among the rows and columns of given matrix A and for the linear forms expressing basic variables in terms of free variables. Therefore, instead of LR and UR, matrices C and H, containing linear combinations, are returned.

Observe carefully that the calculated factorization belongs to the interchanged matrix A^r. Therefore, we use $A^r \cdot X^r = R^r$ instead of $A \cdot X = R$.

Let X^r, R^r be partitioned into $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ and $\begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$.

Then, from $A^r \cdot X^r = R^r$ is obtained:

$$\begin{pmatrix} L \\ LR \end{pmatrix} \cdot (U,\ UR) \cdot \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$$

More explicitly:

$$L \cdot U \cdot X_1 + L \cdot UR \cdot X_2 = R_1$$

$$LR \cdot U \cdot X_1 + LR \cdot UR \cdot X_2 = R_2$$

Since L and U are nonsingular, this implies that:

$$X_1 = U^{-1} \cdot L^{-1} R_1 - U^{-1} \cdot UR \cdot X_2$$

$$R_2 = LR \cdot L^{-1} \cdot R_1$$

For the user's convenience:

LR is replaced by $C_1 = LR \cdot L^{-1}$

UR is replaced by $H = -U^{-1} \cdot UR$

while L and U remain unchanged.

For consistency it is necessary to set $R_2 = C_1 \cdot R_1$ and to obtain homogeneous solutions from the equation:

$$X_1 = H \cdot X_2$$

In case of a consistent system of equations $A^r \cdot X^r = R^r$, the general solution is:

$$X^r = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \text{with } X_1 = U^{-1} \cdot L^{-1} \cdot R_1 + H \cdot X_2$$

while the values of the free variables contained in X_2 may be chosen arbitrarily.

Programming Considerations:

Let a_{ik} be the absolutely greatest element of the original matrix A, which is found first in column-wise scan. The internal tolerance TOL is set equal to $|EPS \cdot a_{ik}|$.

If, at the m-th elimination step, the absolutely greatest element of $D^{m-1}_{m,m}$ is less than or equal to TOL, the submatrix $D^{m-1}_{m,m}$ is interpreted as being the zero matrix. Then m-1 is returned as rank of the given matrix A and further factorization is bypassed.

The calculated factorization belongs to the interchanged matrix A^r. Therefore, we deal with $A^r \cdot X^r = R^r$ instead of $A \cdot X = R$, where:

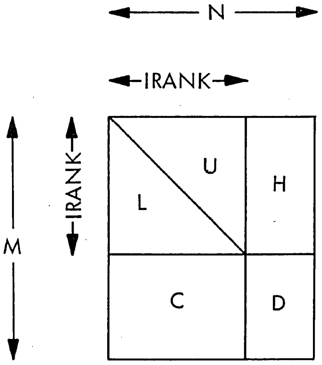
$$\left\{\begin{matrix} X^r \\ R^r \end{matrix}\right\} \text{ is obtained from } \left\{\begin{matrix} X \\ R \end{matrix}\right\} \text{ using the}$$

$$\left\{\begin{matrix} ICOL(k) \\ IROW(i) \end{matrix}\right\} \text{ element of } \left\{\begin{matrix} X \\ R \end{matrix}\right\} \text{ as } \left\{\begin{matrix} k\text{-th} \\ i\text{-th} \end{matrix}\right\} \text{ element of}$$

$$\left\{\begin{matrix} X^r \\ R^r \end{matrix}\right\}$$

with $k = 1, 2, \ldots, N$ and $i = 1, 2, \ldots, M$.

Within the storage area originally occupied by the input matrix A, procedure MFGR returns, in a compact scheme, the matrices L, U, C, H, and D (see diagram).



Numerical example

$$\text{Let } A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 4 \\ 2 & 4 & 2 \\ 1 & 4 & -1 \end{pmatrix}, \quad EPS = 1E\text{-}5$$

Procedure MFGR returns L, U, C, H, and D:

$$L = \begin{pmatrix} 1 & 0 \\ 0.5 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & 2 \\ 0 & 3 \end{pmatrix},$$

$$C = \begin{pmatrix} 0.5 & 0 \\ 1.5 & -1 \end{pmatrix}, \quad H = \begin{pmatrix} -0.33333325 \\ -0.33333331 \end{pmatrix}, \quad D = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

and combines them in the following compact scheme:

$$\begin{pmatrix} 4 & 2 & -0.33333325 \\ 0.5 & 3 & -0.33333331 \\ 0.5 & 0 & 0 \\ 1.5 & -1 & 0 \end{pmatrix} \text{ and } \begin{matrix} IRANK = 2 \\ IROW = (3, 2, 1, 4) \\ ICOL = (2, 3, 1) \end{matrix}$$

From information in C, IRANK, IROW we get the linear dependencies among rows:

$$\begin{aligned} row(1) &= 0.5 \cdot row(3) + 0 \cdot row(2) \\ row(4) &= 1.5 \cdot row(3) - 1 \cdot row(2) \end{aligned}$$

From information in H, IRANK, ICOL we get the homogeneous solution of $A \cdot X = 0$: $X_1 = H \cdot X_2$:

$$\begin{aligned} x_2 &= -0.33333325 \; x_1 \\ x_3 &= -0.33333331 \; x_1 \end{aligned}$$

and with

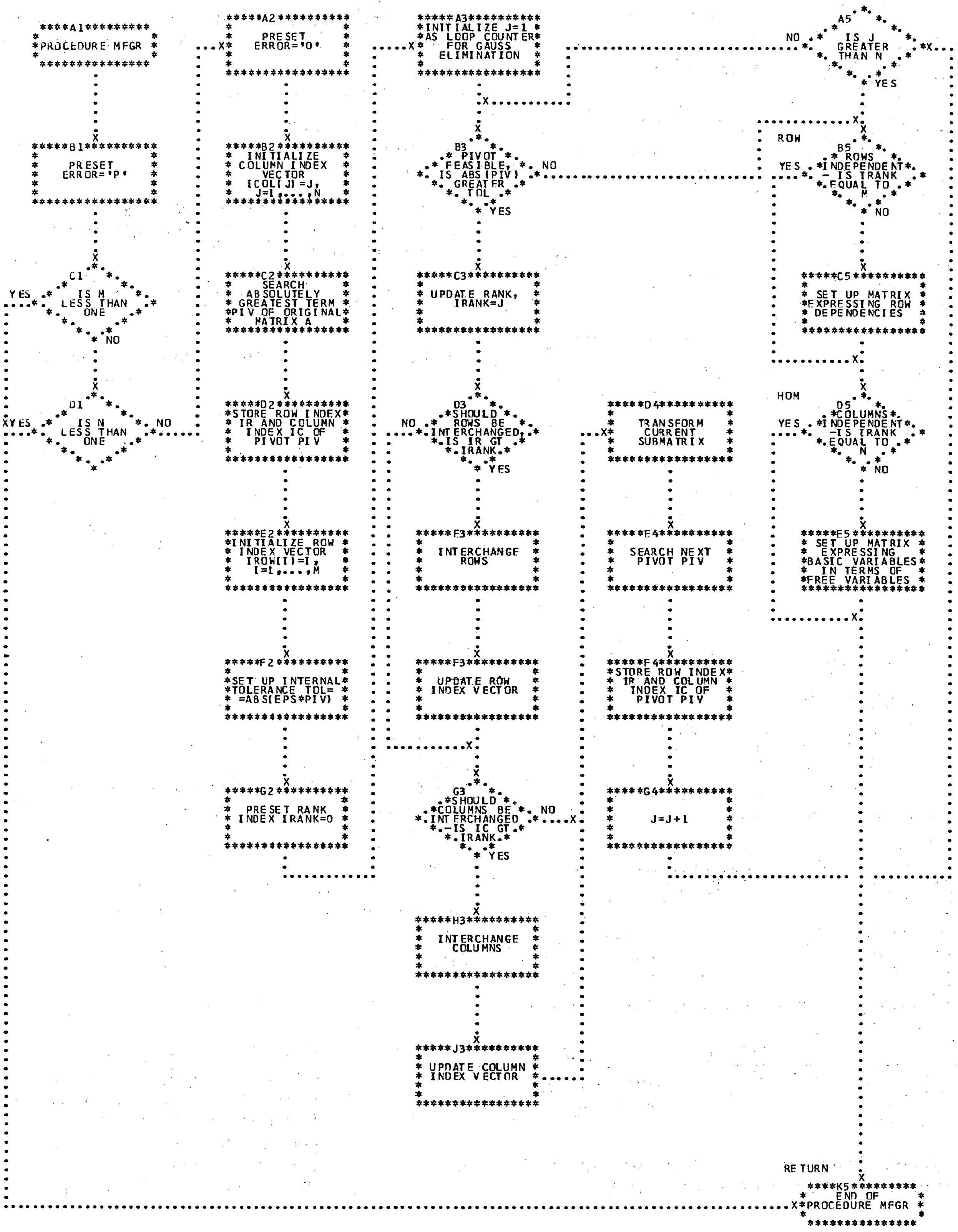
column (1) $\cdot x_1$ + column (2) $\cdot x_2$ + column (3) $\cdot x_3 = 0$, the linear dependencies among columns:

column (1) = 0.33333325 · column (2) +0.33333331 · column (3).

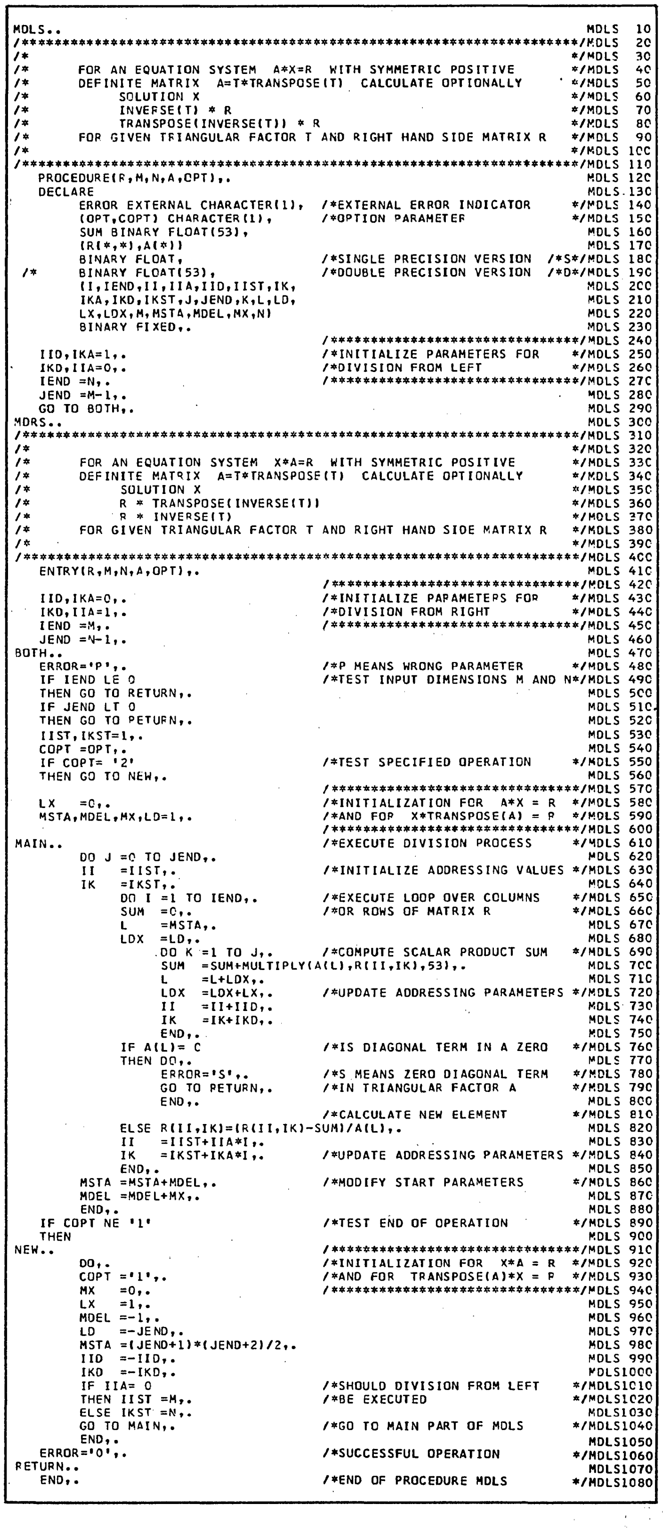
Multiplying the triangular factors L, U we get:

$$L \cdot U = \begin{pmatrix} a_{32} & a_{33} \\ a_{22} & a_{23} \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}$$

FOR A GIVEN GENERAL RECTANGULAR MATRIX MFGR PERFORMS THE FOLLOWING
DETERMINES RANK AND LINEARLY INDEPENDENT ROWS AND COLUMNS (BASIS), FACTORIZES A SUBMATRIX OF MAXIMAL RANK,
EXPRESSES NONBASIC ROWS IN TERMS OF BASIC ROWS, EXPRESSES BASIC VARIABLES IN TERMS OF FREE VARIABLES.


A1
PROCEDURE MFGR
B1
PRESET ERROR='P'
C1
IS M LESS THAN ONE
YES
NO
D1
IS N LESS THAN ONE
YES
NO
A2
PRESET ERROR='0'
B2
INITIALIZE COLUMN INDEX VECTOR ICOL(J)=J, J=1,...,N
C2
SEARCH ABSOLUTELY GREATEST TERM PIV OF ORIGINAL MATRIX A
D2
STORE ROW INDEX IR AND COLUMN INDEX IC OF PIVOT PIV
E2
INITIALIZE ROW INDEX VECTOR IROW(I)=I, I=1,...,M
F2
SET UP INTERNAL TOLERANCE TOL= =ABS(EPS*PIV)
G2
PRESET RANK INDEX IRANK=0
A3
INITIALIZE J=1 AS LOOP COUNTER FOR GAUSS ELIMINATION
B3
PIVOT FEASIBLE, IS ABS(PIV) GREATER TOL
NO
YES
C3
UPDATE RANK, IRANK=J
D3
SHOULD ROWS BE INTERCHANGED, IS IR GT IRANK
NO
YES
E3
INTERCHANGE ROWS
F3
UPDATE ROW INDEX VECTOR
G3
SHOULD COLUMNS BE INTERCHANGED -IS IC GT IRANK
NO
YES
H3
INTERCHANGE COLUMNS
J3
UPDATE COLUMN INDEX VECTOR
D4
TRANSFORM CURRENT SUBMATRIX
E4
SEARCH NEXT PIVOT PIV
F4
STORE ROW INDEX IR AND COLUMN INDEX IC OF PIVOT PIV
G4
J=J+1
A5
IS J GREATER THAN N
NO
YES
ROW
B5
ROWS INDEPENDENT - IS IRANK EQUAL TO M
YES
NO
C5
SET UP MATRIX EXPRESSING ROW DEPENDENCIES
HOM
D5
COLUMNS INDEPENDENT -IS IRANK EQUAL TO N
YES
NO
E5
SET UP MATRIX EXPRESSING BASIC VARIABLES IN TERMS OF FREE VARIABLES
RETURN
K5
END OF PROCEDURE MFGR

- <u>Subroutine</u> MDLS/MDRS



```
MDLS..
/**********************************************************************/
/*                                                                    */
/*      FOR AN EQUATION SYSTEM  A*X=R  WITH SYMMETRIC POSITIVE        */
/*      DEFINITE MATRIX  A=T*TRANSPOSE(T)  CALCULATE OPTIONALLY       */
/*           SOLUTION X                                               */
/*           INVERSE(T) * R                                           */
/*           TRANSPOSE(INVERSE(T)) * R                                */
/*      FOR GIVEN TRIANGULAR FACTOR T AND RIGHT HAND SIDE MATRIX R    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(R,M,N,A,OPT),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */
        (OPT,COPT) CHARACTER(1),      /*OPTION PARAMETER              */
        SUM BINARY FLOAT(53),
        (R(*,*),A(*))
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (I,IEND,II,IIA,IID,IIST,IK,
        IKA,IKD,IKST,J,JEND,K,L,LD,
        LX,LDX,M,MSTA,MDEL,MX,N)
        BINARY FIXED,.
                                      /********************************/
   IID,IKA=1,.                        /*INITIALIZE PARAMETERS FOR     */
   IKD,IIA=0,.                        /*DIVISION FROM LEFT            */
   IEND =N,.                          /********************************/
   JEND =M-1,.
   GO TO BOTH,.
MDRS..
/**********************************************************************/
/*                                                                    */
/*      FOR AN EQUATION SYSTEM  X*A=R  WITH SYMMETRIC POSITIVE        */
/*      DEFINITE MATRIX  A=T*TRANSPOSE(T)  CALCULATE OPTIONALLY       */
/*           SOLUTION X                                               */
/*           R * TRANSPOSE(INVERSE(T))                                */
/*           R * INVERSE(T)                                           */
/*      FOR GIVEN TRIANGULAR FACTOR T AND RIGHT HAND SIDE MATRIX R    */
/*                                                                    */
/**********************************************************************/
   ENTRY(R,M,N,A,OPT),.
                                      /********************************/
   IID,IKA=0,.                        /*INITIALIZE PARAMETERS FOR     */
   IKD,IIA=1,.                        /*DIVISION FROM RIGHT           */
   IEND =M,.                          /********************************/
   JEND =N-1,.
BOTH..
   ERROR='P',.                        /*P MEANS WRONG PARAMETER       */
   IF IEND LE 0                       /*TEST INPUT DIMENSIONS M AND N*/
   THEN GO TO RETURN,.
   IF JEND LT 0
   THEN GO TO PETURN,.
   IIST,IKST=1,.
   COPT =OPT,.
   IF COPT= '2'                       /*TEST SPECIFIED OPERATION      */
   THEN GO TO NEW,.
                                      /********************************/
   LX   =0,.                          /*INITIALIZATION FOR  A*X = R   */
   MSTA,MDEL,MX,LD=1,.                /*AND FOP  X*TRANSPOSE(A) = P   */
                                      /********************************/
MAIN..                                /*EXECUTE DIVISION PROCESS      */
        DO J =0 TO JEND,.
        II   =IIST,.                  /*INITIALIZE ADDRESSING VALUES */
        IK   =IKST,.
             DO I =1 TO IEND,.        /*EXECUTE LOOP OVER COLUMNS     */
             SUM  =0,.                /*OR ROWS OF MATRIX R           */
             L    =MSTA,.
             LDX  =LD,.
                  DO K =1 TO J,.      /*COMPUTE SCALAR PRODUCT SUM    */
                  SUM  =SUM+MULTIPLY(A(L),R(II,IK),53),.
                  L    =L+LDX,.
                  LDX  =LDX+LX,.      /*UPDATE ADDRESSING PARAMETERS */
                  II   =II+IID,.
                  IK   =IK+IKD,.
                  END,.
             IF A(L)= 0               /*IS DIAGONAL TERM IN A ZERO    */
             THEN DO,.
                  ERROR='S',.         /*S MEANS ZERO DIAGONAL TERM    */
                  GO TO PETURN,.      /*IN TRIANGULAR FACTOR A        */
                  END,.
                                      /*CALCULATE NEW ELEMENT         */
             ELSE R(II,IK)=(R(II,IK)-SUM)/A(L),.
             II   =IIST+IIA*I,.
             IK   =IKST+IKA*I,.       /*UPDATE ADDRESSING PARAMETERS */
             END,.
        MSTA =MSTA+MDEL,.             /*MODIFY START PARAMETERS       */
        MDEL =MDEL+MX,.
        END,.
   IF COPT NE '1'                     /*TEST END OF OPERATION         */
   THEN
NEW..                                 /********************************/
        DO,.                          /*INITIALIZATION FOR  X*A = R   */
        COPT ='1',.                   /*AND FOR  TRANSPOSE(A)*X = P   */
        MX   =0,.                     /********************************/
        LX   =1,.
        MDEL =-1,.
        LD   =-JEND,.
        MSTA =(JEND+1)*(JEND+2)/2,.
        IID  =-IID,.
        IKD  =-IKD,.
        IF IIA= 0                     /*SHOULD DIVISION FROM LEFT     */
        THEN IIST =M,.                /*BE EXECUTED                   */
        ELSE IKST =N,.
        GO TO MAIN,.                  /*GO TO MAIN PART OF MDLS       */
        END,.
   ERROR='0',.                        /*SUCCESSFUL OPERATION          */
RETURN..
   END,.                              /*END OF PROCEDURE MDLS         */
```

Purpose:

For a system of equations AX = R with symmetric positive definite matrix A = T • T^T, MDLS performs the following calculations depending on the character of the input parameter OPT:

OPT = '1'	R is replaced by $T^{-1} \cdot R$
OPT = '2'	R is replaced by $(T^{-1})^T \cdot R$
otherwise	R is replaced by $(T \circ T^T)^{-1} \circ R$

Usage:

CALL MDLS (R, M, N, A, OPT);

R(M,N) -	BINARY FLOAT [(53)] Given general right-hand-side matrix with M rows and N columns. Resultant solution depending on the option parameter OPT.
M -	BINARY FIXED Given number of rows of matrix R and the order of matrix A.
N -	BINARY FIXED Given number of columns of matrix R.
A(M*(M+1)/2) -	BINARY FLOAT [(53)] Given one-dimensional array containing lower triangular matrix T stored rowwise in compressed form (possibly resultant array A of SSP procedure MFS).
OPT -	CHARACTER (1) Given option parameter for selection of operation. (See "Purpose" above.)

Purpose:

For a system of equations XA = R with symmetric positive definite matrix A = T • T^T, MDRS performs the following calculations, depending on the character of an input parameter OPT:

OPT = '1'	R is replaced by $R \cdot (T^{-1})^T$
OPT = '2'	R is replaced by $R \cdot T^{-1}$
otherwise	R is replaced by $R \cdot (T \cdot T^T)^{-1}$

Usage:

CALL MDRS (R, M, N, A, OPT);

R(M, N) -	BINARY FLOAT [(53)] Given general right-hand-side matrix with M rows and N columns. Resultant solution depending on the option parameter OPT.
M -	BINARY FIXED Given number of rows of matrix R

N - BINARY FIXED
Given number of columns of matrix R and the order of matrix A.

A(N*(N+1)/2) - BINARY FLOAT [(53)]
Given one-dimensional array containing lower triangular matrix T stored rowwise in compressed form (possibly resultant array A of SSP procedure MFS).

OPT - CHARACTER (1)
Given option parameter for selection of operation (see "Purpose", above).

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='P' - means error in specified dimensions: $M \le 0$ and/or $N \le 0$

ERROR='S' - means given triangular factor T has at least one diagonal term (pivot) equal to zero -- that is, matrix A is not positive definite.

The given lower triangular factor T is assumed to be stored in compressed form, that is, rowwise in successive K*(K+1)/2 storage locations, where K is the number of rows (or columns) implied by compatibility:

K = M in procedure MDLS
K = N in procedure MDRS

During calculation the lower triangular matrix T is not changed. The right-hand-side matrix R is replaced by the solution depending on the character of parameter OPT.

Method:

It is supposed that the symmetric positive definite matrix A is given in the factored form (Cholesky):

$$A = T \cdot T^T$$

where T is the lower triangular factor (possibly calculated by SSP procedure MFS) and T^T the transpose of T.

The required calculations are done using forward and/or backward substitutions.

Mathematical Background:

Calculation of $X = T^{-1} \cdot R$ is done using forward substitution to obtain X from $T \cdot X = R$.

Calculation of $Y = (T^{-1})^T \cdot R$ is done using backward substitution to obtain Y from $T^T \cdot Y = R$.

Calculation of $Z = (T \cdot T^T)^{-1} \cdot R$ is done by first solving $T \cdot X = R$ and then solving $T^T \cdot Z = X$.

Calculation of $X = R(T^{-1})^T$ is done using forward substitution to obtain X from $X \cdot T^T = R$.

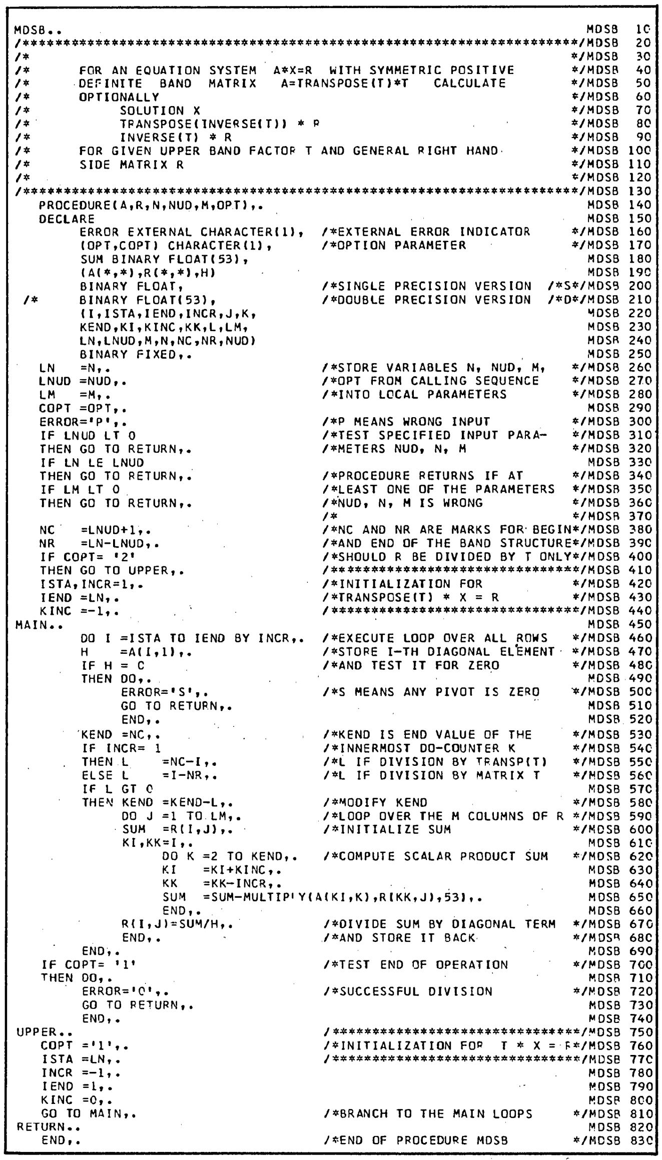
Calculation of $Y = R \cdot T^{-1}$ is done using backward substitution to obtain Y from $Y \cdot T = R$.

Calculation of $Z = R \cdot (T \cdot T^T)^{-1}$ is done by first solving $X \cdot T^T = R$ and then solving $Z \cdot T = X$.

Programming Considerations:

The given lower triangular matrix T is assumed to be stored rowwise in successive storage locations. During calculation, T is not changed, while the right-hand-side matrix R is replaced by the solution depending on parameter OPT. If any diagonal element (pivot) of T is zero, the error parameter ERROR is set to 'S' and further calculation is bypassed. Any zero pivot in T means that the matrix $A = T \cdot T^T$ is not positive definite, possibly because of severe loss of significance in the factorization routine.

• Subroutine MDSB



```
MDSB..
/********************************************************************/
/*                                                                  */
/*      FOR AN EQUATION SYSTEM  A*X=R  WITH SYMMETRIC POSITIVE     */
/*      DEFINITE  BAND  MATRIX   A=TRANSPOSE(T)*T   CALCULATE       */
/*      OPTIONALLY                                                  */
/*           SOLUTION X                                             */
/*           TRANSPOSE(INVERSE(T)) * R                              */
/*           INVERSE(T) * R                                         */
/*      FOR GIVEN UPPER BAND FACTOR T AND GENERAL RIGHT HAND        */
/*      SIDE MATRIX R                                               */
/*                                                                  */
/********************************************************************/
   PROCEDURE(A,R,N,NUD,M,OPT),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR    */
        (OPT,COPT) CHARACTER(1),      /*OPTION PARAMETER            */
        SUM BINARY FLOAT(53),
        (A(*,*),R(*,*),H)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION /*D*/
        (I,ISTA,IEND,INCR,J,K,
        KEND,KI,KINC,KK,L,LM,
        LN,LNUD,M,N,NC,NR,NUD)
        BINARY FIXED,.
   LN   =N,.                          /*STORE VARIABLES N, NUD, M,  */
   LNUD =NUD,.                        /*OPT FROM CALLING SEQUENCE   */
   LM   =M,.                          /*INTO LOCAL PARAMETERS       */
   COPT =OPT,.
   ERROR='P',.                        /*P MEANS WRONG INPUT         */
   IF LNUD LT 0                       /*TEST SPECIFIED INPUT PARA-  */
   THEN GO TO RETURN,.                /*METERS NUD, N, M            */
   IF LN LE LNUD
   THEN GO TO RETURN,.                /*PROCEDURE RETURNS IF AT     */
   IF LM LT 0                         /*LEAST ONE OF THE PARAMETERS */
   THEN GO TO RETURN,.                /*NUD, N, M IS WRONG          */
                                      /*                            */
   NC   =LNUD+1,.                     /*NC AND NR ARE MARKS FOR BEGIN*/
   NR   =LN-LNUD,.                    /*AND END OF THE BAND STRUCTURE*/
   IF COPT= '2'                       /*SHOULD R BE DIVIDED BY T ONLY*/
   THEN GO TO UPPER,.                 /******************************/
   ISTA,INCR=1,.                      /*INITIALIZATION FOR          */
   IEND =LN,.                         /*TRANSPOSE(T) * X = R        */
   KINC =-1,.                         /******************************/
MAIN..
        DO I =ISTA TO IEND BY INCR,.  /*EXECUTE LOOP OVER ALL ROWS  */
        H    =A(I,1),.                /*STORE I-TH DIAGONAL ELEMENT */
        IF H = 0                      /*AND TEST IT FOR ZERO        */
        THEN DO,.
             ERROR='S',.              /*S MEANS ANY PIVOT IS ZERO   */
             GO TO RETURN,.
             END,.
        KEND =NC,.                    /*KEND IS END VALUE OF THE    */
        IF INCR= 1                    /*INNERMOST DO-COUNTER K      */
        THEN L    =NC-I,.             /*L IF DIVISION BY TRANSP(T)  */
        ELSE L    =I-NR,.             /*L IF DIVISION BY MATRIX T   */
        IF L GT 0
        THEN KEND =KEND-L,.           /*MODIFY KEND                 */
             DO J =1 TO LM,.          /*LOOP OVER THE M COLUMNS OF R */
             SUM  =R(I,J),.           /*INITIALIZE SUM              */
             KI,KK=I,.
                  DO K =2 TO KEND,.   /*COMPUTE SCALAR PRODUCT SUM  */
                  KI   =KI+KINC,.
                  KK   =KK-INCR,.
                  SUM  =SUM-MULTIPLY(A(KI,K),R(KK,J),53),.
                  END,.
             R(I,J)=SUM/H,.           /*DIVIDE SUM BY DIAGONAL TERM */
             END,.                    /*AND STORE IT BACK           */
        END,.
   IF COPT= '1'                       /*TEST END OF OPERATION       */
   THEN DO,.
        ERROR='0',.                   /*SUCCESSFUL DIVISION         */
        GO TO RETURN,.
        END,.
UPPER..                               /******************************/
   COPT ='1',.                        /*INITIALIZATION FOR  T * X = R*/
   ISTA =LN,.                         /******************************/
   INCR =-1,.
   IEND =1,.
   KINC =0,.
   GO TO MAIN,.                       /*BRANCH TO THE MAIN LOOPS    */
RETURN..
   END,.                              /*END OF PROCEDURE MDSB       */
```

Purpose:

Depending on the character of the input parameter OPT, MDSB performs the following operations on a system of equations $A \cdot X = R$ with symmetric positive definite band matrix:

$$A = T^T \cdot T$$

OPT = '1' R is replaced by $(T^{-1})^T \cdot R$

OPT = '2' R is replaced by $T^{-1} \cdot R$

otherwise R is replaced by $(T^T \cdot T)^{-1} \cdot R$

Usage:

CALL MDSB (A, R, N, NUD, M, OPT);

A(N, NUD+1) - BINARY FLOAT [(53)]
Given two-dimensional array containing the upper band factor T stored rowwise such that A(i, 1) are the diagonal elements (i = 1, 2, ... N). This could be the resultant array A from SSP procedure MFSB.

R(N, M) - BINARY FLOAT [(53)]
Given general right-hand-side matrix with N rows and M columns. Resultant solution depending on option parameter OPT.

N - BINARY FIXED
Given number of rows of matrices R and A.

NUD - BINARY FIXED
Given number of upper codiagonals of symmetric matrix A.

M - BINARY FIXED
Given number of columns of matrix R.

OPT - CHARACTER (1)
Given option parameter for selection of operation (see "Purpose").

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='P' - Indicates an error in specified dimension: NUD <0 or N ≤ NUD

ERROR='S' - means the given band factor T has at least one diagonal term (pivot) equal to zero -- that is, matrix A is not positive definite.

Upper factor matrix T, consisting of main diagonal and NUD upper codiagonals, is assumed to be stored rowwise in array A(N, NUD+1) such that A(i, 1) are the diagonal elements of T (i=1,2,...,N). SSP procedure MFSB provides upper band factor T in its resultant array A, which may be used directly for input in MDSB.

During calculation in MDSB, the band matrix T is not changed. The right-hand-side matrix R is replaced by a solution depending on the input character of parameter OPT. Input values N and NUD should satisfy the restriction

$$0 \leq \text{NUD} < \text{N}$$

Method:

Depending on the actual character of OPT, division of R by T^T and/or T is performed using forward and/or backward substitutions. The result is returned in the locations of R.

For reference see:

R. S. Martin and J. H. Wilkinson, "Solution of Symmetric and Unsymmetric Band Equations and the Calculation of Eigenvectors of Band Matrices", Numerische Mathematik, Vol. 9, iss. 4, 1967, pp. 279-301.

H. Rutishauser, "Algorithmus 1-Lineares Gleichungssystem mit symmetrischer positiv-definiter Bandmatrix nach Cholesky", Computing (Archives for Electronic Computing), Vol. 1, iss. 1, 1966, pp. 77-78.

Mathematical Background:

The given elements of the upper factor matrix T are to be stored rowwise in array A so that A(i, 1) are the diagonal elements of T (i = 1, 2, ..., N).

Calculation of $X = (T^{-1})^T \cdot R$ is done using forward substitution to obtain X from $T^T \cdot X = R$ and satisfying the following recursive scheme:

$$x_{ik} = \frac{1}{a_{ik}}\left[r_{ik} - \sum_{m=m_0}^{i-1} a_{m,i+1-m} \cdot x_{mk}\right]$$

$$m_0 = \max(1,\ i - \text{NUD})\ ; \quad \begin{array}{l} i = 1, 2, \dots, N \\ k = 1, 2, \dots, M \end{array}$$

(Any symbol $\sum_{m=m_0}^{r} c_m$ is to be interpreted as zero if $r < m_0$.)

After each x_{ik} is computed, it is stored in the location r_{ik}. Analogously, computing $Y = T^{-1} \cdot R$ is the same as solving the equation $T \cdot Y = R$ for Y. This is done using backward substitution in a similar recursive scheme:

$$y_{ik} = \frac{1}{a_{ik}}\left[r_{ik} - \sum_{m=2}^{m_0} a_{im} \cdot y_{i-1+m,k}\right]$$

$$m_0 = \min(\text{NUD} + 1,\ \ N + 1 - i)$$

$$\begin{array}{l} i = N,\ N-1, \dots, 1 \\ k = 1,\ 2, \dots, M \end{array}$$

Calculation of $Z = A^{-1} \cdot R = (T^T \cdot T)^{-1} \cdot R$ is done by first computing X from $T^T \cdot X = R$ and overwriting on R, then solving $T \cdot Z = X$, again in the locations of R. If R is equal to the unit matrix, this process replaces R with the inverse A^{-1} of A. It should be noted that in general A^{-1} is no longer a band matrix.

Programming Considerations:

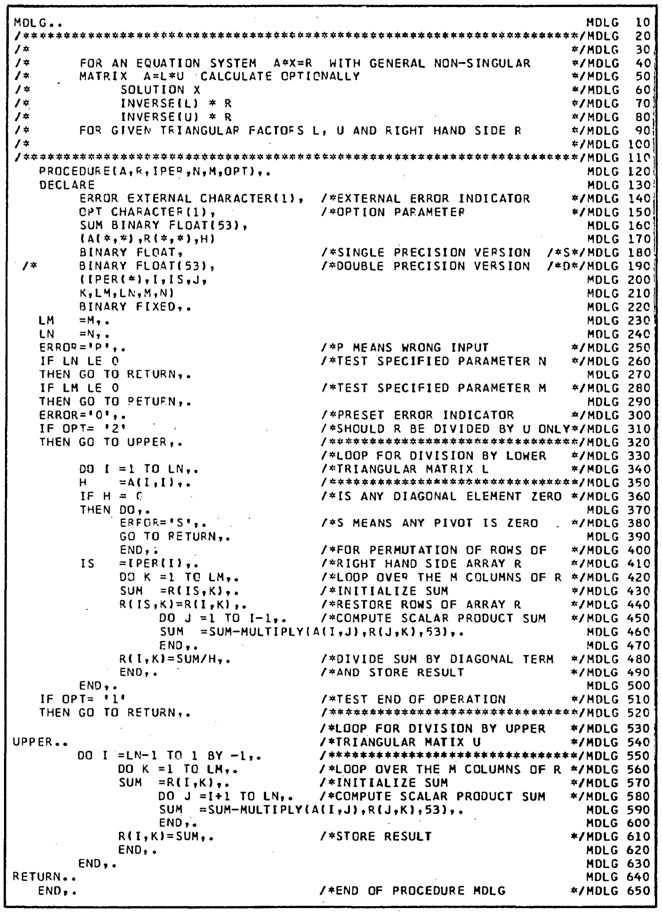
The upper band factor matrix T is assumed to be stored rowwise in the two-dimensional array A(N, NUD+1) such that A(i, 1) are the diagonal elements of T (i = 1, 2, ..., N). Therefore, the elements A(i,k) of array A with i + k > N are irrelevant and not used within MDSB.

During calculation, the upper band factor T is not changed, while the right-hand-side matrix R is replaced by a solution depending on the character of parameter OPT.

If any diagonal element A(i, 1) of factor T is zero, the error parameter ERROR is set to 'S' and further calculation is bypassed. Any zero pivot of T means that matrix $A = T^T \cdot T$ is not positive definite. This is possibly due to severe loss of significance in the factorization routine.

If the SSP procedure MFSB provides the factor matrix T directly as input for MDSB, the resultant error indicator ERROR from MFSB should be tested.

- Subroutine MDLG



```
MDLG..
/**********************************************************************/
/*                                                                    */
/*      FOR AN EQUATION SYSTEM  A*X=R  WITH GENERAL NON-SINGULAR      */
/*      MATRIX  A=L*U  CALCULATE OPTIONALLY                           */
/*           SOLUTION X                                               */
/*           INVERSE(L) * R                                           */
/*           INVERSE(U) * R                                           */
/*      FOR GIVEN TRIANGULAR FACTORS L, U AND RIGHT HAND SIDE R       */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,R,IPER,N,M,OPT),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */
        OPT CHARACTER(1),             /*OPTION PARAMETER              */
        SUM BINARY FLOAT(53),
        (A(*,*),R(*,*),H)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (IPER(*),I,IS,J,
        K,LM,LN,M,N)
        BINARY FIXED,.
   LM   =M,.
   LN   =N,.
   ERROR='P',.                        /*P MEANS WRONG INPUT           */
   IF LN LE 0                         /*TEST SPECIFIED PARAMETER N    */
   THEN GO TO RETURN,.
   IF LM LE 0                         /*TEST SPECIFIED PARAMETER M    */
   THEN GO TO RETURN,.
   ERROR='0',.                        /*PRESET ERROR INDICATOR        */
   IF OPT= '2'                        /*SHOULD R BE DIVIDED BY U ONLY*/
   THEN GO TO UPPER,.                 /******************************/
                                      /*LOOP FOR DIVISION BY LOWER    */
        DO I =1 TO LN,.               /*TRIANGULAR MATRIX L           */
        H    =A(I,I),.                /******************************/
        IF H = 0                      /*IS ANY DIAGONAL ELEMENT ZERO */
        THEN DO,.
             ERROR='S',.              /*S MEANS ANY PIVOT IS ZERO     */
             GO TO RETURN,.
             END,.                    /*FOR PERMUTATION OF ROWS OF    */
        IS   =IPER(I),.               /*RIGHT HAND SIDE ARRAY R       */
             DO K =1 TO LM,.          /*LOOP OVER THE M COLUMNS OF R */
             SUM  =R(IS,K),.          /*INITIALIZE SUM                */
             R(IS,K)=R(I,K),.         /*RESTORE ROWS OF ARRAY R       */
                  DO J =1 TO I-1,.    /*COMPUTE SCALAR PRODUCT SUM    */
                  SUM  =SUM-MULTIPLY(A(I,J),R(J,K),53),.
                  END,.
             R(I,K)=SUM/H,.           /*DIVIDE SUM BY DIAGONAL TERM   */
             END,.                    /*AND STORE RESULT              */
        END,.
   IF OPT= '1'                        /*TEST END OF OPERATION         */
   THEN GO TO RETURN,.                /******************************/
                                      /*LOOP FOR DIVISION BY UPPER    */
UPPER..                               /*TRIANGULAR MATIX U            */
        DO I =LN-1 TO 1 BY -1,.       /******************************/
             DO K =1 TO LM,.          /*LOOP OVER THE M COLUMNS OF R */
             SUM  =R(I,K),.           /*INITIALIZE SUM                */
                  DO J =I+1 TO LN,.   /*COMPUTE SCALAR PRODUCT SUM    */
                  SUM  =SUM-MULTIPLY(A(I,J),R(J,K),53),.
                  END,.
             R(I,K)=SUM,.             /*STORE RESULT                  */
             END,.
        END,.
RETURN..
   END,.                              /*END OF PROCEDURE MDLG         */
```

Purpose:

For a system of equations A• X = R, where A = L • U is a general nonsingular matrix, MDLG performs the following calculations, depending on the character of an input parameter OPT:

OPT = '1'	R is replaced by $L^{-1} \cdot R$
OPT = '2'	R is replaced by $U^{-1} \cdot R$
otherwise	R is replaced by $(L \cdot U)^{-1} \cdot R$

Usage:

CALL MDLG (A, R, IPER, N, M, OPT);

A(N,N) - BINARY FLOAT [(53)]
Given two-dimensional array containing lower and upper triangular matrices L and U where the unit diagonal of U is omitted.

R(N,M) - BINARY FLOAT [(53)]
Given general right-hand-side matrix with N rows and M columns.
Resultant solution depending on the option parameter OPT.

IPER(N) - BINARY FIXED
Given integer vector containing the permutations of rows of the matrix A in factorization steps.

N - BINARY FIXED
Given order of matrix A and number of rows of matrix R.

M - BINARY FIXED
Given number of columns of matrix R.

OPT - CHARACTER (1)
Given option parameter for selection of operation (see "Purpose").

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='P' - means error in specified dimensions: $M \le 0$ and/or $N \le 0$

ERROR='S' - means that a diagonal element (pivot) in the given lower triangular matrix L is zero; further calculation is bypassed.

The given matrix A is assumed to be factorized into a product of a lower triangular matrix L and an upper triangular matrix U using partial pivoting with row interchanges, where L and U are overwritten on A, omitting the unit diagonal of U. Details of the row interchanges are to be stored in the vector IPER. This required factorization may be obtained using the SSP procedure MFG. The resulting arrays A and IPER are used as input for MDLG.

During calculation in MDLG the arrays A and IPER are not changed. The right-hand-side matrix R is replaced by a solution depending on the character of parameter OPT.

Method:

The required calculations are performed using forward and/or backward substitutions, where the interchange information is combined with the lower triangular matrix L.

Mathematical Background:

Suppose a general nonsingular matrix A of order n is factored into the form:

$$A = P \cdot L \cdot U$$

where L is the lower triangular matrix, U the upper triangular matrix with unit diagonal, and P the permutation matrix corresponding to the integer vector

IPER. Then $X = L^{-1} \cdot P^{-1} \cdot R = L^{-1} \cdot \bar{R}$ is calculated using forward substitution to obtain X from $L \cdot X = P^{-1} \cdot R = \bar{R}$. $\bar{R}$ is obtained from R by interchanging rows in the same way as the rows of matrix A are interchanged during partial pivoting in any factorization routine (for example, MFG).

To calculate $Y = U^{-1} \cdot R$ backward substitution is used in obtaining Y from $U \cdot Y = R$. Calculation of $Z = U^{-1} \cdot L^{-1} \cdot P^{-1} \cdot R = U^{-1} \cdot L^{-1} \cdot \bar{R}$ is done by first solving $L \cdot X = \bar{R}$ and then solving $U \cdot Z = X$.

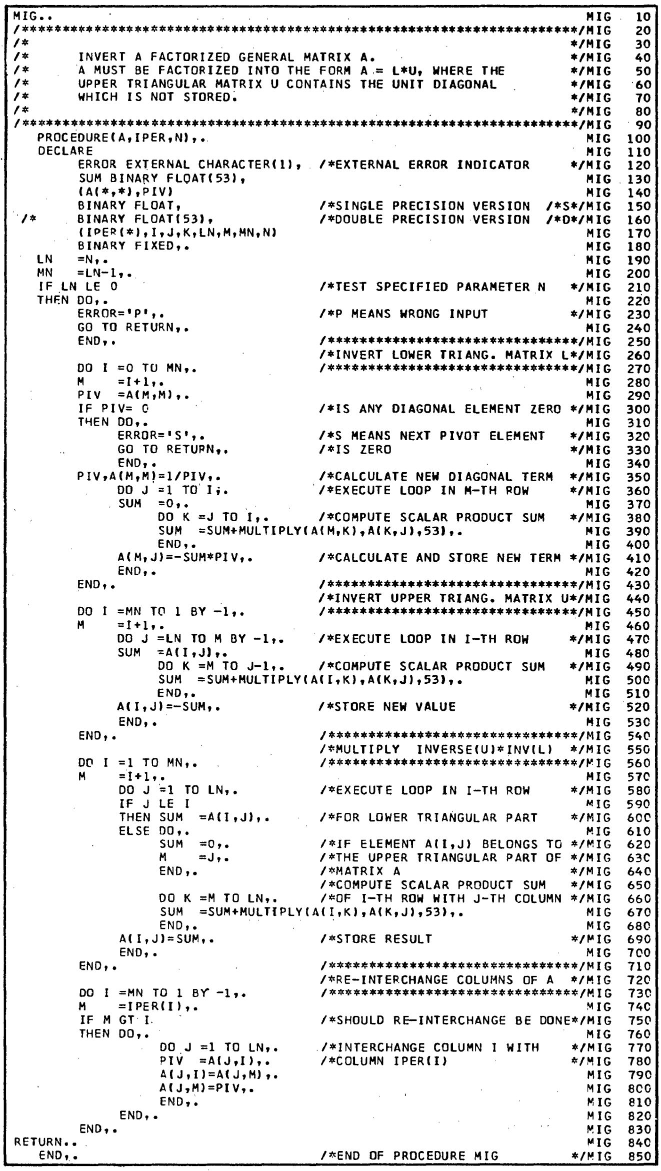
Programming Considerations:

Matrix A is assumed to be given in the factored form:

$$A = P \cdot L \cdot U$$

where the lower triangular matrix L and the upper triangular matrix U are overwritten on A, omitting the unit diagonal of U. The permutation matrix P is obtained by interchanging the rows of an n by n unit matrix according to information stored in the vector IPER.

- <u>Subroutine</u> MIG



```
MIG..
/**********************************************************************/
/*                                                                    */
/*      INVERT A FACTORIZED GENERAL MATRIX A.                         */
/*      A MUST BE FACTORIZED INTO THE FORM A = L*U, WHERE THE         */
/*      UPPER TRIANGULAR MATRIX U CONTAINS THE UNIT DIAGONAL          */
/*      WHICH IS NOT STORED.                                          */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,IPER,N),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */
        SUM BINARY FLOAT(53),
        (A(*,*),PIV)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (IPER(*),I,J,K,LN,M,MN,N)
        BINARY FIXED,.
   LN   =N,.
   MN   =LN-1,.
   IF LN LE 0                         /*TEST SPECIFIED PARAMETER N    */
   THEN DO,.
        ERROR='P',.                   /*P MEANS WRONG INPUT           */
        GO TO RETURN,.
        END,.                         /********************************/
                                      /*INVERT LOWER TRIANG. MATRIX L*/
        DO I =0 TO MN,.               /********************************/
        M    =I+1,.
        PIV  =A(M,M),.
        IF PIV= 0                     /*IS ANY DIAGONAL ELEMENT ZERO */
        THEN DO,.
             ERROR='S',.              /*S MEANS NEXT PIVOT ELEMENT    */
             GO TO RETURN,.           /*IS ZERO                       */
             END,.
        PIV,A(M,M)=1/PIV,.            /*CALCULATE NEW DIAGONAL TERM   */
             DO J =1 TO I;.           /*EXECUTE LOOP IN M-TH ROW      */
             SUM  =0,.
                  DO K =J TO I,.      /*COMPUTE SCALAR PRODUCT SUM    */
                  SUM  =SUM+MULTIPLY(A(M,K),A(K,J),53),.
                  END,.
             A(M,J)=-SUM*PIV,.        /*CALCULATE AND STORE NEW TERM */
             END,.
        END,.                         /********************************/
                                      /*INVERT UPPER TRIANG. MATRIX U*/
        DO I =MN TO 1 BY -1,.         /********************************/
        M    =I+1,.
             DO J =LN TO M BY -1,.    /*EXECUTE LOOP IN I-TH ROW      */
             SUM  =A(I,J),.
                  DO K =M TO J-1,.    /*COMPUTE SCALAR PRODUCT SUM    */
                  SUM  =SUM+MULTIPLY(A(I,K),A(K,J),53),.
                  END,.
             A(I,J)=-SUM,.            /*STORE NEW VALUE               */
             END,.
        END,.                         /********************************/
                                      /*MULTIPLY  INVERSE(U)*INV(L)   */
        DO I =1 TO MN,.               /********************************/
        M    =I+1,.
             DO J =1 TO LN,.          /*EXECUTE LOOP IN I-TH ROW      */
             IF J LE I
             THEN SUM  =A(I,J),.      /*FOR LOWER TRIANGULAR PART     */
             ELSE DO,.
                  SUM  =0,.           /*IF ELEMENT A(I,J) BELONGS TO */
                  M    =J,.           /*THE UPPER TRIANGULAR PART OF */
                  END,.               /*MATRIX A                      */
                                      /*COMPUTE SCALAR PRODUCT SUM    */
                  DO K =M TO LN,.     /*OF I-TH ROW WITH J-TH COLUMN */
                  SUM  =SUM+MULTIPLY(A(I,K),A(K,J),53),.
                  END,.
             A(I,J)=SUM,.             /*STORE RESULT                  */
             END,.
        END,.                         /********************************/
                                      /*RE-INTERCHANGE COLUMNS OF A   */
        DO I =MN TO 1 BY -1,.         /********************************/
        M    =IPER(I),.
        IF M GT I                     /*SHOULD RE-INTERCHANGE BE DONE*/
        THEN DO,.
                  DO J =1 TO LN,.     /*INTERCHANGE COLUMN I WITH     */
                  PIV  =A(J,I),.      /*COLUMN IPER(I)                */
                  A(J,I)=A(J,M),.
                  A(J,M)=PIV,.
                  END,.
             END,.
        END,.
RETURN..
   END,.                              /*END OF PROCEDURE MIG          */
```

Purpose:

MIG inverts a general nonsingular matrix A, which is given in the factored form:

$$A = L \cdot U$$

where the upper triangular matrix U contains the unit diagonal, which is not stored.

Usage:

CALL MIG (A, IPER, N);

A(N, N) - BINARY FLOAT [(53)]
Given two-dimensional array containing lower and upper triangular factors L and

U, where the unit diagonal of U is not stored (possibly resultant array A of SSP procedure MFG).
Resultant calculated inverse of matrix A.

IPER(N) - BINARY FIXED
Given vector contains the permutations of rows of the matrix in factorization steps.

N - BINARY FIXED
Given order of matrix A.

Remarks:

ERROR='P' - means error in specified dimension: $N \leq 0$

ERROR='S' - means that a diagonal element (pivot) in the given lower triangular matrix L is zero; further calculation is bypassed.

Method:

It is required that the general nonsingular matrix A be given in the factored form:

$$A = L \cdot U$$

where L means the lower triangular matrix and U the upper triangular matrix with unit diagonal. L and the superdiagonal part of U are stored in the storage locations of A, which may be factored by SSP procedure MFG.

In the first step MIG inverts L, giving L^{-1}, which is overwritten on L. In the second step U^{-1} is calculated and stored in U. Then U^{-1} is multiplied by L^{-1}, giving, in an order determined by pivoting, the columns of A^{-1}. These, finally, are reordered to produce A^{-1}.

For reference see:

A. S. Householder, The Theory of Matrices in Numerical Analysis, 1965, pp. 125-130.
A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, Vol. 2, 1967, pp. 69-71.
R. Zurmühl, Matrizen, 1964, pp. 75-77.

Mathematical Background:

Suppose A, a general nonsingular matrix of order N, is factored into the form:

$$A = P \cdot L \cdot U$$

where L is the lower triangular matrix, U the upper triangular matrix with unit diagonal, and P the row-permutation matrix (unit matrix with interchanged rows) resulting from partial pivoting in any factorization routine. Then A^{-1} is calculated in four steps:

1. The elements $\bar{l}_{ik}$ of L^{-1} are computed from the elements l_{ik} of L with the following recursive formulas:

$$\bar{l}_{ik} = -\frac{1}{l_{ii}} \sum_{m=k}^{i-1} l_{im} \cdot \bar{l}_{mk} \qquad i>k$$

$$\bar{l}_{ik} = \frac{1}{l_{ii}} \qquad i=k$$

$$\bar{l}_{ik} = 0 \qquad i<k$$

2. The elements $\bar{u}_{ik}$ of U^{-1} are computed from the elements u_{ik} of U with the following recurrsive formulas:

$$\bar{u}_{ik} = -u_{ik} - \sum_{m=i+1}^{k-1} u_{im} \cdot \bar{u}_{mk} \qquad i<k$$

(any symbol $\sum_{m=k}^{k-1} x_m$ is to be interpreted as zero)

$$\bar{u}_{ik} = 1 \qquad i=k$$

$$\bar{u}_{ik} = 0 \qquad i>k$$

3. The elements $\bar{a}_{ik}$ of the product $U^{-1} \cdot L^{-1}$ are computed with the formulas:

$$\bar{a}_{ik} = \bar{l}_{ik} + \sum_{m=i+1}^{N} \bar{u}_{im} \cdot \bar{l}_{mk} \qquad i \geq k$$

$$\bar{a}_{ik} = \sum_{m=k}^{N} \bar{u}_{im} \cdot \bar{l}_{mk} \qquad i<k$$

4. The resultant product $U^{-1} \cdot L^{-1}$ is multiplied on the right by the inverse permutation matrix P^{-1} giving:

$$A^{-1} = U^{-1} \cdot L^{-1} \cdot P^{-1}$$

That is, the columns of the product $U^{-1} \cdot L^{-1}$ are rearranged according to the interchanges performed during the factorization of the matrix.

Programming Considerations:

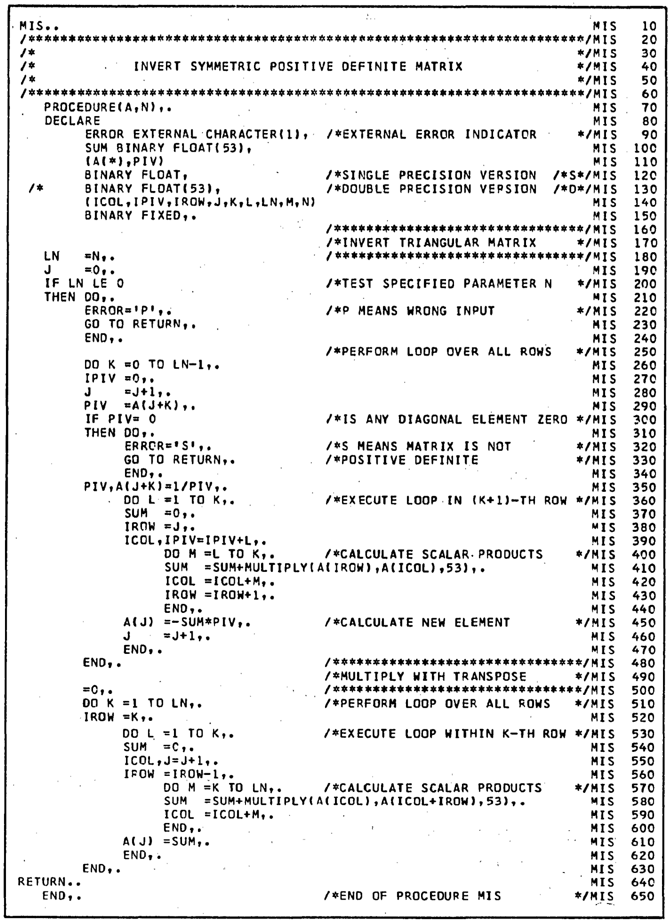
Matrix A is required in the factored form:

A = P · L · U

where L is the lower triangular matrix, U the upper triangular matrix with unit diagonal, and P the permutation matrix corresponding to the integer vector IPER. L and the superdiagonal part of U are to be stored in the two-dimensional array A.

If the required factorization is done using the SSP procedure MFG, the resulting arrays A and IPER may be directly used as input for MIG. The inverse matrix A^{-1} is calculated by MIG in the storage locations of array A.

- <u>Subroutine</u> MIS



```
MIS..
/**************************************************************************/
/*                                                                        */
/*          INVERT SYMMETRIC POSITIVE DEFINITE MATRIX                     */
/*                                                                        */
/**************************************************************************/
   PROCEDURE(A,N),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),   /*EXTERNAL ERROR INDICATOR         */
        SUM BINARY FLOAT(53),
        (A(*),PIV)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION    /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VEPSION    /*D*/
        (ICOL,IPIV,IROW,J,K,L,LN,M,N)
        BINARY FIXED,.
                                       /**********************************/
                                       /*INVERT TRIANGULAR MATRIX         */
   LN   =N,.                           /**********************************/
   J    =0,.
   IF LN LE 0                          /*TEST SPECIFIED PARAMETER N       */
   THEN DO,.
        ERROR='P',.                    /*P MEANS WRONG INPUT              */
        GO TO RETURN,.
        END,.
                                       /*PERFORM LOOP OVER ALL ROWS       */
        DO K =0 TO LN-1,.
        IPIV =0,.
        J    =J+1,.
        PIV  =A(J+K),.
        IF PIV= 0                      /*IS ANY DIAGONAL ELEMENT ZERO     */
        THEN DO,.
             ERRCR='S',.               /*S MEANS MATRIX IS NOT            */
             GO TO RETURN,.            /*POSITIVE DEFINITE                */
             END,.
        PIV,A(J+K)=1/PIV,.
             DO L =1 TO K,.            /*EXECUTE LOOP IN (K+1)-TH ROW     */
             SUM  =0,.
             IROW =J,.
             ICOL,IPIV=IPIV+L,.
                  DO M =L TO K,.       /*CALCULATE SCALAR PRODUCTS        */
                  SUM  =SUM+MULTIPLY(A(IROW),A(ICOL),53),.
                  ICOL =ICOL+M,.
                  IROW =IROW+1,.
                  END,.
             A(J) =-SUM*PIV,.          /*CALCULATE NEW ELEMENT            */
             J    =J+1,.
             END,.
        END,.                          /**********************************/
                                       /*MULTIPLY WITH TRANSPOSE          */
        =C,.                           /**********************************/
        DO K =1 TO LN,.                /*PERFORM LOOP OVER ALL ROWS       */
        IROW =K,.
             DO L =1 TO K,.            /*EXECUTE LOOP WITHIN K-TH ROW     */
             SUM  =C,.
             ICOL,J=J+1,.
             IFOW =IROW-1,.
                  DO M =K TO LN,.      /*CALCULATE SCALAR PRODUCTS        */
                  SUM  =SUM+MULTIPLY(A(ICOL),A(ICOL+IROW),53),.
                  ICOL =ICOL+M,.
                  END,.
             A(J) =SUM,.
             END,.
        END,.
RETURN..
   END,.                               /*END OF PROCEDURE MIS             */
```

Purpose:

MIS inverts a symmetric positive definite matrix A, which is given in factored form (Cholesky):

A = T · transpose (T)

Usage:

CALL MIS (A, N);

A(N*(N+1)/2) - BINARY FLOAT [(53)]
Given one-dimensional array containing the lower triangular factor T of matrix A stored rowwise in compressed form (possibly resultant array A of SSP procedure MFS). Resultant lower triangular part of calculated inverse (A) stored rowwise in compressed form.

N - BINARY FIXED
Given order of matrices A and T.

Remarks:

ERROR='P' means error in specified dimension: $N \leq 0$

ERROR='S' means given triangular factor T has at least one pivot equal to zero -- that is, matrix A is not positive definite.

The given lower triangular factor T is assumed to be stored in compressed form -- that is, rowwise in N*(N+1)/2 successive storage locations. On return the lower triangular part of the inverse of A is stored in the same way.

Method:

It is supposed that the symmetric positive definite matrix A is given in the factored form (Cholesky):

$$A = T \cdot \text{transpose}\ (T)$$

where T is the lower triangular factor, possibly calculated by SSP procedure MIS.

In the first step MIS inverts the given triangular matrix T in the storage locations of T. Using

$$\text{inverse}\ (\text{transpose}\ (T)) = \text{transpose}\ (\text{inverse}\ (T))$$

in the second step MIS multiplies inverse (T) with its transpose on the same storage locations, giving

$$\text{inverse}\ (A) = \text{transpose}\ (\text{inverse}\ (T)) \cdot \text{inverse}\ (T)$$

Thus, the given lower triangular factor T is replaced by the lower part of the resultant inverse (A).

For reference see:

A. S. Householder, The Theory of Matrices in Numerical Analysis, 1965, pp. 125-130.
R. Zurmühl, Matrizen, 1964, pp. 77-79.

Mathematical Background:

Suppose the symmetric positive definite matrix A is factored in the form:

$$A = T \cdot \text{transpose}\ (T)$$

where T is a lower triangular factor matrix. Then:

$$\text{inverse}\ (A) = \text{transpose}\ (\text{inverse}\ (T)) \cdot \text{inverse}\ (T)$$

1. The elements $\bar{t}_{ik}$ of inverse (T) are computed from the elements t_{ik} of T using the following recursive formulas:

$$\bar{t}_{ik} = -\frac{\sum_{m=k}^{i-1} \bar{t}_{mk} \cdot t_{im}}{t_{ii}} \qquad i>k$$

$$\bar{t}_{ik} = \frac{1}{t_{ii}} \qquad i=k$$

$$\bar{t}_{ik} = 0 \qquad i<k$$

2. From inverse (T) the elements $\bar{a}_{ik}$ of inverse (A) are calculated as follows:

$$\bar{a}_{ik} = \sum_{m=i}^{N} \bar{t}_{mk} \cdot \bar{t}_{mi} \qquad i \geq k$$

with $\bar{a}_{ik} = \bar{a}_{ki}$

Programming Considerations:

The given lower triangular matrix T is assumed to be stored in compressed form -- that is, rowwise in N • (N+1)/2 successive storage locations. The lower triangular part of the resultant inverse (A) is returned in these locations of T.

If any pivot of the input matrix T is equal to zero, the error parameter ERROR is set to 'S' and further calculation is bypassed. Any zero pivot in T means that matrix A = T • transpose (T) is not positive definite, possibly because of severe loss of significance in the factorization routine.

Subroutine MINV

```
MINV..
/**********************************************************************/
/*                                                                    */
/*      TO INVERT A MATRIX                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,N,D,CON),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),
        (I,J,K,N,L(N),M(N))
        FIXED BINARY,
        (A(*,*),BIGA,HOLD,D,CON,S)
        BINARY FLOAT,.                /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.           /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
   ERROR='0',.
   IF N LE 0
   THEN DO,.
        ERROR='1',.                   /* ORDER OF MATRIX = 0.         */
        GO TO FIN,.
        END,.
   IF CON= 0
   THEN S    =1.0E-5,.                /* SINGLE PRECISION VERSION /*S*/
 /*THEN S    =1.0E-15,.               /* DOUBLE PRECISION VERSION /*D*/
   ELSE S    =CON,.
   IF N = 1                           /* INVERT A SCALAR              */
   THEN DO,.
        D    =A(1,1),.
        IF ABS(D) LE S
        THEN DO,.
             ERROR='2',.
             END,.
        ELSE A(1,1) = 1/D,.
        GO TO FIN,.
        END,.
   D    =1.0,.                        /* SEARCH FOR LARGEST ELEMENT   */
        DO K = 1 TO N,.
        L(K) =K,.
        M(K) =K,.
        BIGA =A(K,K),.
             DO I=K TO N,.
                  DO J=K TO N,.
                  IF ABS(BIGA) LT ABS(A(I,J))
                  THEN DO,.
                       BIGA =A(I,J),.
                       L(K) =I,.
                       M(K) =J,.
                       END,.
                  END,.
             END,.
        J    =L(K),.                  /* INTERCHANGE ROWS             */
        IF L(K) GT K
        THEN DO,.
                  DO I = 1 TO N,.
                  HOLD =-A(K,I),.
                  A(K,I)=A(J,I),.
                  A(J,I)=HOLD,.
                  END,.
             END,.
        I    =M(K),.                  /* INTERCHANGE COLUMNS          */
        IF M(K) GT K
        THEN DO,.
                  DO J = 1 TO N,.
                  HOLD =-A(J,K),.
                  A(J,K)=A(J,I),.
                  A(J,I)=HOLD,.
                  END,.
             END,.
        IF ABS(BIGA) LE S
        THEN DO,.
             D    =0.0,.
             GO TO COMP,.
             END,.
/*                                                                    */
/*      DIVIDE COLUMNS BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS      */
/*      CONTAINED IN BIGA)                                            */
             DO I = 1 TO N,.
             IF I NE K
             THEN A(I,K)=A(I,K)/(-A(K,K)),.
             END,.
             DO I = 1 TO N,.          /* REDUCE MATRIX                */
             IF I NE K
             THEN DO,.
                       DO J = 1 TO N,.
                       IF J NE K
                       THEN A(I,J)=A(I,K)*A(K,J)+A(I,J),.
                       END,.
                  END,.
             END,.
             DO J = 1 TO N,.
             IF J NE K                /* DIVIDE BY ROW PIVOT          */
             THEN A(K,J)=A(K,J)/A(K,K),.
             END,.
        D    =D*A(K,K),.              /* COMPUTE DETERMINANT          */
COMP..
        IF ABS(D) LE S
        THEN DO,.
             ERROR='2',.              /* DETERMINANT IS ZERO          */
             GO TO FIN,.
             END,.
        A(K,K)=1.0/A(K,K),.           /* REPLACE PIVOT BY RECIPROCAL */
        END,.
/*                                                                    */
/*      FINAL ROW AND COLUMN INTERCHANGE                              */
/*                                                                    */
   K    =N,.
LOOP..
   K    =K-1,.
   IF K GT 0
   THEN DO,.
        I    =L(K),.
        IF I GT K
        THEN DO,.
                  DO J = 1 TO N,.
                  HOLD =A(J,K),.
                  A(J,K)=-A(J,I),.
                  A(J,I)=HOLD,.
                  END,.
             END,.
        J    =M(K),.
        IF J GT K
        THEN DO,.
                  DO I = 1 TO N,.
                  HOLD =A(K,I),.
                  A(K,I)=-A(J,I),.
                  A(J,I)=HOLD,.
                  END,.
             END,.
        GO TO LOOP,.
        END,.
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE MINV         */
```

Purpose:

MINV inverts a general square matrix.

Usage:

CALL MINV (A, N, D, CON);

A(N, N) - BINARY FLOAT [(53)]
Given matrix.
Resultant inverse of given matrix.

N - BINARY FIXED
Given order of matrix A.

D - BINARY FLOAT [(53)]
Resultant determinant.

CON - BINARY FLOAT [(53)]
Given constant with which the determinant is compared. If the given value of CON is zero, the program assigns the value 10^{-5} in single precision and 10^{-15} is double precision.

Remarks:

A must be a general square matrix.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - means that the order of the matrix is less than or equal to zero.

ERROR=2 - means that the absolute value of the determinant is less than or equal to the specified constant CON (see description of parameters for explanation).

Method:

The standard Gauss-Jordan method is used and the determinant is calculated.

- Subroutine MLSQ

```
MLSQ..                                                                 MLSQ   10
/**********************************************************************/MLSQ   20
/*                                                                   */MLSQ   30
/*     LINEAR LEAST SQUARES PROBLEM SOLVED USING HOUSEHOLDER TRANSF.*/MLSQ   40
/*                                                                   */MLSQ   50
/**********************************************************************/MLSQ   60
   PROCEDURE(A,B,M,N,K),.                                              MLSQ   70
   DECLARE                                                             MLSQ   80
      (A(*,*),B(*,*),PIVR,MAXA)                                        MLSQ   90
      BINARY FLOAT,                    /*SINGLE PRECISION VERSION  /*S*/MLSQ 100
 /*   BINARY FLOAT(53),                /*DOUBLE PRECISION VERSION  /*D*/MLSQ 110
      (AUX(N),H,SIG,BETA)                                              MLSQ 120
      BINARY FLOAT(53),                                                MLSQ 130
      (TOL,PIV(N))                                                     MLSQ 140
      BINARY FLOAT,                                                    MLSQ 150
      ERROR EXTERNAL CHARACTER(1),     /*EXTERNAL ERROR INDICATOR    */MLSQ 160
      (I,J,K,L,M,N,PIVI,LM,LN,LK)                                      MLSQ 170
      BINARY FIXED,.                                                   MLSQ 180
   LM   =M,.                                                           MLSQ 190
   LN   =N,.                                                           MLSQ 200
   LK   =K,.                                                           MLSQ 210
   SIG  =0,.                                                           MLSQ 220
   ERROR='D',.                         /*PRESET ERROR INDICATOR      */MLSQ 230
   IF LM GE LN                         /*IF M LESS THAN N            */MLSQ 240
   THEN IF LN GE 0                     /*OR IF N NOT POSITIVE        */MLSQ 250
   THEN IF LK GT 0                     /*OR IF K NOT POSITIVE        */MLSQ 260
   THEN DO,.                           /*THEN BYPASS OPERATION       */MLSQ 270
         DO L = 1 TO LN,.              /*CALCULATE SCALARPRODUCTS OF */MLSQ 280
         H    =0,.                     /*COLUMNS                     */MLSQ 290
            DO I = 1 TO LM,.                                           MLSQ 300
            H    =H+MULTIPLY(A(I,L),A(I,L),53),.                       MLSQ 310
            END,.                                                      MLSQ 320
         IF H GE SIG                                                   MLSQ 330
         THEN DO,.                                                     MLSQ 340
            SIG  =H,.                  /*SAVE MAXIMAL SCALARPRODUCT  */MLSQ 350
            PIVI =L,.                  /*SAVE SUBSCRIPT OF PIVOTCOLUMN*/MLSQ 360
            END,.                                                      MLSQ 370
         AUX(L),PIV(L)=H,.                                             MLSQ 380
         END,.                         /*****************************/MLSQ 390
                                       /*DECOMPOSITION LOOP          */MLSQ 400
      ERROR='0',.                      /*****************************/MLSQ 410
         DO L = 1 TO LN,.                                              MLSQ 420
         TOL  =PIV(PIVI),.             /*ORIGINAL LENGTH OF PIVOTCOL. */MLSQ 430
         IF PIVI GT L                  /*SHOULD COLUMN BE INTERCHANGED*/MLSQ 440
         THEN DO..                                                     MLSQ 450
            H    =AUX(L),.             /*INTERCHANGE SCALARPRODUCTS  */MLSQ 460
            AUX(L)=AUX(PIVI),.                                         MLSQ 470
            PIV(PIVI)=PIV(L)..                                         MLSQ 480
            AUX(PIVI)=H,.                                              MLSQ 490
               DO J=L TO LM,. /*INTERCHANGE LOWER PART OF            */MLSQ 500
               PIVR =A(J,L),. /*COLUMNS OF A                         */MLSQ 510
               A(J,L)=A(J,PIVI),.                                      MLSQ 520
               A(J,PIVI)=PIVP,.                                        MLSQ 530
               END,.                                                   MLSQ 540
            END,.                                                      MLSQ 550
         IF L GT 1                     /*RECALCULATE COLUMN LENGTH   */MLSQ 560
         THEN DO,.                     /*TO AVOID ROUND-OFF PROBLEMS */MLSQ 570
            SIG  =0,.                                                  MLSQ 580
               DO I = L TO LM,.                                        MLSQ 590
               SIG  =SIG+MULTIPLY(A(I,L),A(I,L),53),.                  MLSQ 600
               END,.                                                   MLSQ 610
            END,.                                                      MLSQ 620
         IF TOL= 0                                                     MLSQ 630
         THEN DO,.                                                     MLSQ 640
            IF ERROR NE 'B'                                            MLSQ 650
            THEN IF ERROR NE 'W'                                       MLSQ 660
            THEN ERROR='S',.           /*GIVEN A HAS ZERO-COLUMN(S)  */MLSQ 670
            ELSE ERROR='B',.                                           MLSQ 680
            TOL  =1,.                                                  MLSQ 690
            END,.                                                      MLSQ 700
         BETA =TOL*1E-10,.             /*SINGLE PRECISION VERSION  /*S*/MLSQ 710
 /*      BETA =TOL*1E-20,.             /*DOUBLE PRECISION VERSION  /*D*/MLSQ 720
         IF SIG LE BETA                                                MLSQ 730
         THEN DO,.                     /*INDICATE LOSS OF SIGNIFICANCE*/MLSQ 740
            IF ERROR NE 'B'                                            MLSQ 750
            THEN IF ERROR NE 'S'                                       MLSQ 760
            THEN ERROR='W',.                                           MLSQ 770
            ELSE ERROR='B',.                                           MLSQ 780
            IF SIG LE 0                                                MLSQ 790
            THEN SIG  =BETA,.          /*MODIFY ZERO VALUE           */MLSQ 800
            END,.                                                      MLSQ 810
         SIG  =SQRT(SIG),.                                             MLSQ 820
         H    =A(L,L),.                                                MLSQ 830
         IF H LT 0                                                     MLSQ 840
         THEN SIG  =-SIG,.             /*FORCE SIGN(SIG) TO SIGN(H)  */MLSQ 850
         PIV(L)=PIVI,.                 /*SAVE INTERCHANGE INFORMATION */MLSQ 860
         A(L,L),BETA=H+SIG,.           /*TRANSFORM DIAGONAL ELEMENT  */MLSQ 870
         AUX(L)=-SIG,.                 /*SAVE DIAGONAL ELEMENT       */MLSQ 880
         BETA =SIG*BETA,.                                              MLSQ 890
                                       /*TRANSFORM SUBMATRIX OF A    */MLSQ 900
         PIVR =0,.                                                     MLSQ 910
            DO J = L+1 TO LN,.  /*TRANSFORM LOWER PART OF A          */MLSQ 920
            H    =0,.           /*COLUMNS L+1 UP TO N ONLY           */MLSQ 930
               DO I = L TO LM,.                                        MLSQ 940
               H    =H+MULTIPLY(A(I,L),A(I,J),53),.                    MLSQ 950
               END,.                                                   MLSQ 960
            SIG  =H/BETA,.      /*MODIFY J-TH COLUMN                 */MLSQ 970
               DO I = LM TO L BY -1,.                                  MLSQ 980
               H    =A(I,J),.                                          MLSQ 990
               A(I,J)=H-A(I,L)*SIG,.                                   MLSQ1000
               END,.            /*NEXT UPDATE COLUMN LENGTH          */MLSQ1010
            H    =A(L,J),.                                             MLSQ1020
            AUX(J),H=AUX(J)-H*H,.                                      MLSQ1030
            IF H GE PIVR        /*SEARCH NEXT PIVOTCOLUMN            */MLSQ1040
            THEN DO,.                                                  MLSQ1050
               PIVR =H,.                                               MLSQ1060
               PIVI =J,.                                               MLSQ1070
               END,.                                                   MLSQ1080
            END,.                                                      MLSQ1090
                                /*TRANSFORM LOWER PART OF            */MLSQ1100
            DO J = 1 TO LK,.    /*RIGHT HAND SIDE MATRIX B           */MLSQ1110
            H    =0,.                                                  MLSQ1120
               DO I = L TO LM,.                                        MLSQ1130
               H    =H+MULTIPLY(A(I,L),B(I,J),53),.                    MLSQ1140
               END,.                                                   MLSQ1150
            MAXA =H/BETA,.      /*MODIFY J-TH COLUMN                 */MLSQ1160
               DO I = L TO LM,.                                        MLSQ1170
               B(I,J)=B(I,J)-A(I,L)*MAXA,.                             MLSQ1180
               END,.                                                   MLSQ1190
            END,.                                                      MLSQ1200
         END,.                  /*END OF DECOMPOSITION LOOP          */MLSQ1210
                                /************************************/MLSQ1220
         DO J = LN TO 1 BY -1,. /*BACKSUBSTITUTION,INTERCHANGE       */MLSQ1230
            DO I = 1 TO LK,.    /************************************/MLSQ1240
```

```
               H    =B(J,I),.                                          MLSQ1250
                    DO L = J+1 TO LN,.                                 MLSQ1260
                    H    =H-MULTIFLY(A(J,L),B(L,I),53),.               MLSQ1270
                    END,.                                              MLSQ1280
               PIVI =PIV(J),.                                          MLSQ1290
               B(J,I)=B(PIVI,I),.                                      MLSQ1300
               B(PIVI,I)=H/AUX(J),.                                    MLSQ1310
               END,.                                                   MLSQ1320
            END,.                                                      MLSQ1330
         IF LN LT LM                   /*COMPUTE LEAST SQUARES       */MLSQ1340
         THEN DO J = 1 TO LK,.         /*IN CASE OF AN OVERDETERMINED */MLSQ1350
            H    =0,.                  /*EQUATION SYSTEM ONLY        */MLSQ1360
               DO I = LN+1 TO LM,.                                     MLSQ1370
               H    =H+MULTIPLY(B(I,J),B(I,J),53),.                    MLSQ1380
               END,.                                                   MLSQ1390
            B(LM,J)=H,.                                                MLSQ1400
            END,.                                                      MLSQ1410
         END,.                         /*END OF OPERATION            */MLSQ1420
      END,.                            /*END OF PROCEDURE MLSQ       */MLSQ1430
```

Purpose:

MLSQ calculates X satisfying AX=B, that is, the solution of a system of linear equations using Householder transformations. The least squares solution is obtained in case of an overdetermined system of equations.

Usage:

CALL MLSQ (A, B, M, N, K);

A(M, N) - BINARY FLOAT [(53)]
Given coefficient matrix of equation system.
A gets destroyed.

B(M, K) - BINARY FLOAT [(53)]
Given matrix of right-hand sides.
Resultant solution of A· X=B stored in upper N rows of B, and if M>N resultant square sum of residuals for I-th right-hand side stored in elements B(M, I) for I = 1, 2, ..., K.

M - BINARY FIXED
Given number of equations, that is, number of rows of matrices A and B.

N - BINARY FIXED
Given number of unknowns, that is, number of columns of matrix A and number of rows of resultant X, which is overlaid with B.

K - BINARY FIXED
Given number of right-hand sides, that is, number of columns of B.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='D' means incorrect dimension(s); not all of the conditions M ≥ N > 0, K > 0 are satisfied. Operation is bypassed.

ERROR='W' means warning, indicating possible loss of significance in resultant X.

ERROR='S' means A has at least one zero-column. Resultant X is a least squares solution (not necessarily of minimal norm).

ERROR='B' implies both ERROR='S' and ERROR='W'; that is, resultant X is a least squares solution, but possibly affected by loss of significance.

The internal relative tolerance for test on loss of significance is set to 10^{-5} in single precision and to 10^{-10} in double precision. In the single precision version, scalar products are accumulated using double precision arithmetic.

Method:

A is reduced to upper triangular form, using Householder transformations successively. The same sequence of transformations is applied to given right-hand-side matrix B. Solution X is then obtained using backsubstitution.

For reference see:

G. Golub, "Numerical Methods for Solving Linear Least Squares Problems", Numerische Mathematik, vol. 7, 1965, pp. 206-216.

Mathematical Background:

Notation

The transpose of a matrix A is written as A^T. The k^{th} column vector of A is written as $A_{*,k}$ and the i^{th} row vector as $A_{i,*}$. The Euclidean norm of the vector $R = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}$ is abbreviated:

$$\|R\| = \sqrt{R^T R} = \sqrt{\sum_{i=1}^{n} r_i^2}$$

Problem

For a given m by n coefficient matrix A with m ≥ n and an m by k matrix B of right-hand sides, an n by k matrix X must be calculated that solves AX = B in the least squares sense, that is:

$$\left\| B_{*j} - A X_{*j} \right\| = \min, \text{ for } j = 1, 2, \ldots, k$$

The determination of X is based on the reduction of the matrix A to an m by n matrix R of the form

$$R = \begin{pmatrix} U \\ O \end{pmatrix}$$

by means of an orthogonal transformation Q, so that U is an upper trangular matrix of order n.

$$QA = R$$

Then, the given equation AX = B can be solved as follows:

$$\begin{aligned} QAX &= QB \\ RX &= QB \\ X &= [U^{-1}\ O]\ QB \end{aligned}$$

if U is of maximal rank (otherwise, see "Programming Considerations"). It is interesting to note that U is the triangular factor provided by the Cholesky factorization of $A^T A$.

$$A^T A = U^T U$$

Householder's transformations

The reduction of the given matrix A to the matrix R can be achieved by means of a sequence of (n-1) orthogonal transformations the product of which will be Q. This can be written as

$$A^{(0)} = A$$

$$A^{(i)} = P^{(i)} A^{(i-1)}, \quad i = 1, \ldots, n-1$$

where $A^{(i)}$ is supposed to have the same form as R in its first i columns, and where $P^{(i)}$ is an orthogonal matrix. Then:

$$R = A^{(n-1)}$$

Among the possible matrices $P^{(i)}$, let us consider those of the form

$$P^{(i)} = I + \alpha^{(i)} W^{(i)} W^{(i)\,T}$$

where I is the unit matrix and w a vector of order m related to the scalar $\alpha^{(i)} \neq 0$ by

$$< W^{(i)}, W^{(i)} > = -\frac{2}{\alpha^{(i)}}$$

It is easy to see that these matrices are orthogonal and symmetric. By definition of $A^{(i)}$, $P^{(i)}$ can be written as

$$P^{(i)} = I + \frac{1}{g^{(i)}\,(v_i^{(i)} - g^{(i)})}\,(v^{(i)} - g^{(i)} e_i)\,(v^{(i)} - g^{(i)} e_i)^T$$

where:

$$v^{(i)T} = (v_1^{(i)}, v_2^{(i)}, \ldots, v_m^{(i)})$$

$$v_j^{(i)} = 0 \text{ for } j < i$$

$$v_j^{(i)} = a_{ji}^{(i-1)} \text{ for } j \geq i$$

$$g^{(i)} = -\operatorname{sign}(v_i^{(i)}) \left\| v^{(i)} \right\|$$

and where e_i is a vector of order m whose components are zero except for the i-th, which is one.

Actually, neither matrices $P^{(i)}$ nor matrix $Q = P^{(n-1)} \ldots P^{(1)}$ is computed explicitly.

Each column k of $A^{(i)}$, $k = i, \ldots, n$, is calculated from column k of $A^{(i-1)}$ as follows

$$A_{*k}^{(i)} = A_{*k}^{(i-1)} + \frac{1}{g^{(i)}\,(v_i^{(i)} - g^{(i)})} < v^{(i)}$$

$$- g^{(i)} e_{i,}\ A_{*k}^{(i-1)} > (v^{(i)} - g^{(i)} e_i)$$

The columns of matrix B are modified in the same manner.

Pivoting

To keep roundoff errors as small as possible, an interchange of columns is performed before the i-th transformation, so that the i-th column of $A^{(i-1)}$ gets permuted with the k-th for which $\left\| v^{(i)} \right\|$ is maximum. k is determined by:

$$s_k^{(i)} = \max_{i \leq j \leq n} (s_j^{(i)})$$

where:

$$s_j^{(i)} = \sum_{q=i}^{m} \left[a_{qj}^{(i-1)} \right]^2$$

Back substitution

When the matrix is reduced to the triangular form, the solution is obtained by back substitution. The interchange of rows determined by the pivoting is applied to the solution as soon as any component is computed.

Programming Considerations:

The procedure may fail if, at any intermediate step i, no column with nonzero parameter $g^{(i)}$can be found -- that is, if no nonzero main diagonal element in U can be generated. In this case, the rank of the matrix A is less than n. Because of roundoff errors this situation may even occur if the rank of the given matrix A equals n. In order to indicate this ill-conditioned case, with its possible loss of significance, each $|g^{(i)}|$ is compared against a tolerance TOL_i. TOL_i is the product of the norm of the corresponding column in the original matrix A times the internal tolerance EPS (10^{-5} in single precision and 10^{-10} in double precision).

1. If the relative tolerances TOL_i are all positive (no zero columns in original A), then ERROR = 'W' if $|g^{(i)}| > TOL_i$ does not hold true for all $i = 1, 2, \ldots, n$. Zero elements $g^{(i)}$ get replaced by $TOL_i \cdot 10^{-10}$ ($TOL \cdot 10^{-20}$ in double precision).

2. If A has zero columns (corresponding $TOL_i = 0$), then ERROR = 'S'. The corresponding $g^{(i)}$ is set to $1E^{-10}$ or $1E^{-20}$.

3. If cases 1 and 2 occur combined, ERROR = 'B'.

Case 1 indicates possible loss of significance in resultant solution X. Case 2 means that X is a least squares solution but possibly not the uniquely determined one of minimal norm.

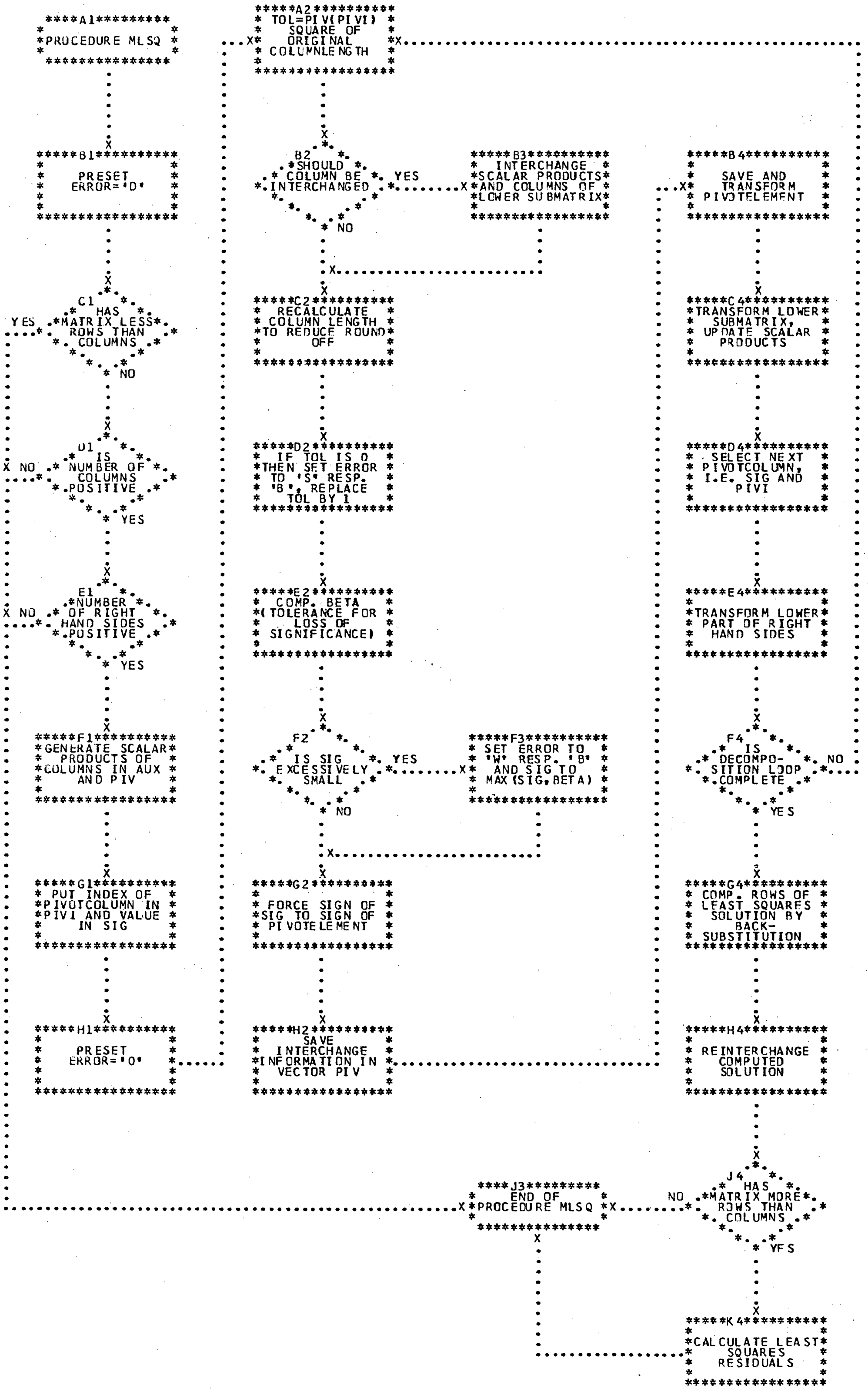
For full understanding of the procedure note tnat:

1. The $g^{(i)}$'s are recalculated to avoid roundoff problems.

2. The resultant X is overlaidwith the given right-hand sides.

3. Least squares deviations are calculated only in case m > n, and stored in the last row of the given right-hand-side matrix.

PROCEDURE MLSQ CALCULATES THE LEAST SQUARES SOLUTICN OF AN OVERDETERMINED SYSTEM OF SIMULTANEOUS LINEAR EQUATIONS


A1
PROCEDURE MLSQ
A2
TOL=PIV(PIVI) SQUARE OF ORIGINAL COLUMNLENGTH
B1
PRESET ERROR='D'
B2
SHOULD COLUMN BE INTERCHANGED
YES
NO
B3
INTERCHANGE SCALAR PRODUCTS AND COLUMNS OF LOWER SUBMATRIX
B4
SAVE AND TRANSFORM PIVOTELEMENT
C1
HAS MATRIX LESS ROWS THAN COLUMNS
YES
NO
C2
RECALCULATE COLUMN LENGTH TO REDUCE ROUND OFF
C4
TRANSFORM LOWER SUBMATRIX, UPDATE SCALAR PRODUCTS
D1
IS NUMBER OF COLUMNS POSITIVE
NO
YES
D2
IF TOL IS 0 THEN SET ERROR TO 'S' RESP. 'B', REPLACE TOL BY 1
D4
SELECT NEXT PIVOTCOLUMN, I.E. SIG AND PIVI
E1
NUMBER OF RIGHT HAND SIDES POSITIVE
NO
YES
E2
COMP. BETA (TOLERANCE FOR LOSS OF SIGNIFICANCE)
E4
TRANSFORM LOWER PART OF RIGHT HAND SIDES
F1
GENERATE SCALAR PRODUCTS OF COLUMNS IN AUX AND PIV
F2
IS SIG EXCESSIVELY SMALL
YES
NO
F3
SET ERROR TO 'W' RESP. 'B' AND SIG TO MAX(SIG,BETA)
F4
IS DECOMPOSITION LOOP COMPLETE
NO
YES
G1
PUT INDEX OF PIVOTCOLUMN IN PIVI AND VALUE IN SIG
G2
FORCE SIGN OF SIG TO SIGN OF PIVOTELEMENT
G4
COMP. ROWS OF LEAST SQUARES SOLUTION BY BACK-SUBSTITUTION
H1
PRESET ERROR='0'
H2
SAVE INTERCHANGE INFORMATION IN VECTOR PIV
H4
REINTERCHANGE COMPUTED SOLUTION
J3
END OF PROCEDURE MLSQ
J4
HAS MATRIX MORE ROWS THAN COLUMNS
NO
YES
K4
CALCULATE LEAST SQUARES RESIDUALS

• Subroutine MGB1/MGB2

```
MGB1..
/**********************************************************************/
/*                                                                    */
/*      FOR AN EQUATION SYSTEM  A*X=P  WITH BAND MATPIX  A=L*U        */
/*      CALCULATE OPTIONALLY                                          */
/*           UPPER TRIANGULAR FACTOR U AND SOLUTION X,                */
/*           UPPER TRIANGULAR FACTOP U AND INVERSE(L)*R,              */
/*           INVERSE(U)*R FOR GIVEN U,R.                              */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(A,P,N,NLD,NUD,M,EPS,OPT),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR     */
        (OPT,COPT) CHARACTER(1),
        EPS BINARY FLOAT,
        SUM BINARY FLOAT(53),
        (A(*,*),R(*,*),L(*),SL(N),PIV,W)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (IPER(*),I,IBAC,IND,INL,IPIV,
        J.K,KL,LM,LLM,LN,LNLD,LNUD,M,
        N,NB,NLD,NUD)
        BINARY FIXED,.
   IND  =1,.
   GO TO BOTH,.
MGB2..
/**********************************************************************/
/*                                                                    */
/*      FOR AN EQUATION SYSTEM  A*X=P  WITH BAND MATRIX  A=L*U        */
/*      COMPUTE OPTIONALLY                                            */
/*           TRIANGULAR FACTORS L,U  POSSIBLY COMBINED WITH           */
/*           CALCULATION OF X OR INVERSE(L)*R,                        */
/*           INVERSE(L)*P OR INVEPSE(A)*P FOR GIVEN L,U,R.            */
/*                                                                    */
/**********************************************************************/
   ENTRY(A,R,L,IPER,N,NLD,NUD,M,EPS,OPT),.
   IND  =2,.
BOTH..
   LN   =N,.                          /*STOPE VAPIABLES N, M, NUD,   */
   LM,LLM=M,.                         /*NLD FROM CALLING SEQUENCE    */
   LNUD =NUD,.                        /*INTO LOCAL PARAMETERS        */
   LNLD =NLD,.
   ERROR='P',.                        /*P MEANS WRONG INPUT          */
   IF LM LE 0                         /*VALUE M MUST BE POSITIVE     */
   THEN GO TO RETURN,.
   IF LNLD LT 0                       /*NUMBER OF LOWER CODIAGONALS  */
   THEN GO TO RETURN,.                /*NLD MAY NOT BE NEGATIVE AND  */
   IF LNLD GE LN                      /*EQUAL TO OR GREATER THAN N   */
   THEN GO TO RETURN,.
   IF LNUD LT C                       /*NUMBER OF UPPER CODIAGONALS  */
   THEN GO TO RETURN,.                /*NUD MAY NOT BE NEGATIVE AND  */
   IF LNUD GE LN                      /*EQUAL TO OR GREATER THAN N   */
   THEN GO TO RETURN,.
   ERROR='0',.                        /*PRESET ERROR INDICATOR       */
   NB   =LNUD+LNLD+1,.                /*CALCULATE THE MAXIMUM WIDTH  */
   IF NB GT LN                        /*OF BAND MATRIX               */
   THEN NB   =LN,.
   IBAC =1,.                          /*IBAC IS AN INDICATOR FOR     */
   KL   =0,.                          /*BACKSUBSTITUTION             */
   COPT =OPT,.
   IF COPT= 'A'                       /*CALCULATE  INVERSE(L) * R    */
   THEN DO,.                          /*FOR GIVEN  L, U, R           */
        IND  =0,.
        IBAC =0,.
        GO TO GAUSS,
        END,.
   IF COPT= 'B'                       /*CALCULATE  INVERSE(U) * R    */
   THEN GO TO BACK,.                  /*FOR GIVEN  U, R              */
   IF COPT= 'C'
   THEN DO,.                          /*CALCULATE  INVERSE(A) * R    */
        IND  =0,.                     /*FOR GIVEN  L, U, R           */
        GO TO GAUSS,.
        END,.
   IF COPT= 'L'                       /*COMPUTE TRIANGULAR FACTOR U  */
   THEN DO,.                          /*AND OPTIONALLY  L  AND       */
        IBAC =0,.                     /*CALCULATE  INVERSE(L) ~ R    */
        GO TO SCAL,.                  /*FOR GIVEN  A, R              */
        END,.
   IF COPT= 'F'                       /*COMPUTE TRIANGULAR FACTORS   */
   THEN DO,.                          /*L AND U  FOR GIVEN MATRIX A  */
        IBAC =0,.
        LLM  =0,.
        GO TO SCAL,.
        END,.                         /*COMPUTE TRIANGULAR FACTOR U  */
   IF COPT= 'U'                       /*AND INVERSE(U)*R FOR GIVEN   */
   THEN LLM  =0,.                     /*A, R                         */
                                      /*                             */
SCAL..                                /*CALCULATE SCALING FACTORS    */
                                      /*                             */
   K    =LNUD,.                       /*K IS AN END INDICATOR FOR    */
   INL  =LNLD+LN-NB+1,.               /*EACH ROW OF MATRIX A         */
   IPIV =NB-LNUD,.
        DO I =1 TO LN,.               /*EXECUTE LOOP OVER ALL ROWS   */
        IF I LE IPIV
        THEN K    =K+1,.              /*IN I-TH ROW THE ELEMENTS     */
        IF I GT INL                   /*A(I,K+1) TO A(I,NB) ARE      */
        THEN K    =K-1,.              /*FILLED UP WITH ZEROS         */
        PIV  =0,.
             DO J =1 TO NB,.          /*EXECUTE LOOP OVER I-TH ROW   */
             IF J GT K
             THEN A(I,J)=0,.          /*FILL UP WITH ZEROS           */
             ELSE DO,.
                  W    =ABS(A(I,J)),. /*COMPUTE ABSOLUTELY GREATEST  */
                  IF W GT PIV         /*ELEMENT PIV IN I-TH ROW OF A */
                  THEN PIV  =W,.
                  END,.
             END,.
        IF PIV= C                     /*TEST FOR ZERO-ROW            */
        THEN DO,.                     /*ALL ELEMENTS IN I-TH ROW OF  */
             ERROR='S',.              /*GIVEN MATRIX A ARE ZERO      */
             GO TO RETURN,.
             END,.                    /*STORE THE RECIPROCAL IN THE  */
        SL(I)=1/PIV,.                 /*VECTOR SL                    */
        END,.                         /**********************************/
GAUSS..                               /*GAUSS ELIMINATION            */
        DO I =1 TO LN-1,.             /**********************************/
        INL  =I+LNLD,.                /*INVERSE(L)*R                 */
        IF INL GT LN
        THEN INL  =LN,.
```

```
        IF IND= C                     /*NO FACTORIZATION             */
        THEN DO,.                     /*CALCULATE INVERSE(L) * R     */
             IPIV =IPER(I),.          /*FOR GIVEN L, U, R            */
             GO TO INTR,.
             END,.
        W    =0,.                     /*INITIALIZE W FOR PIVOTING    */
             DO J =I TO INL,.
             PIV  =ABS(A(J,1))*SL(J),./*MULTIPLY ELEMENTS WITH SCALE */
             IF PIV GT W              /*FACTORS AND SEARCH GREATEST  */
             THEN DO,.                /*PRODUCT                      */
                  W    =PIV,.
                  IPIV =J,.           /*STORE ROW INDEX              */
                  END,.
             END,.
        IF W LE ABS(EPS)              /*TEST FOR LOSS OF SIGNIFICANCE*/
        THEN IF W = 0                 /*AND FOR ZERO                 */
        THEN DO,.
             ERROR='S',.              /*NEXT PIVOT IS ZERO POSSIBLY  */
             GO TO PETURN,.           /*DUE TO LOSS OF SIGNIFICANCE  */
             END,.
        ELSE ERROR='W',.              /*W MEANS WARNING              */
        PIV  =A(IPIV,1),.             /*PIV CONTAINS THE PIVOT       */
        IF IND= 2                     /*STORE INFORMATION FOR ROW-   */
        THEN IPER(I)=IPIV,.           /*PERMUTATIONS                 */
        IF IPIV= I                    /*IS INTERCHANGE NECESSARY     */
        THEN GO TO FSUB,.
        SL(IPIV)=SL(I),.              /*RESTORE SCALING ELEMENTS     */
             DO J =1 TO NB,.
             W    =A(I,J),.           /*INTERCHANGE ROWS IN GIVEN    */
             A(I,J)=A(IPIV,J),.       /*MATRIX A                     */
             A(IPIV,J)=W,.
             END,.
INTR..
             DO J =1 TO LLM,.         /*INTERCHANGE ROWS IN RIGHT    */
             W    =R(I,J),.           /*HAND SIDE MATRIX R           */
             R(I,J)=R(IPIV,J),.
             R(IPIV,J)=W,.
             END,.
FSUB..                                /*MODIFY OPTIONALLY ROWS IN    */
             DO J =I+1 TO INL,.       /*MATRIX A AND IN RIGHT HAND   */
             IF IND= 0                /*SIDE MATRIX R                */
             THEN DO,.
                  KL   =KL+1,.
                  W    =L(KL),.
                  GO TO DIVL,.
                  END,.
             W    =A(J,1)/PIV,.       /*W IS AN ELEMENT OF THE LOWER */
             IF IND= 2                /*TRIANGULAR FACTOR L          */
             THEN DO,.
                  KL   =KL+1,.
                  L(KL)=W,.           /*STORE W INTO L IF REQUESTED  */
                  END,.
                  DO K =2 TO NB,.     /*MODIFY AND SHIFT ROWS OF A   */
                  A(J,K-1)=A(J,K)-W*A(I,K),.
                  END,.
             A(J,NB)=C,.              /*LAST TERM IS SET TO ZERO     */
DIVL..                                /*MODIFY ROWS OF R TO COMPUTE  */
                  DO K =1 TO LLM..    /*INVERSE(L)*R                 */
                  R(J,K)=R(J,K)-W*R(I,K),.
                  END,.
             END,.
        END,.
   IF IND= 2
   THEN IPER(LN)=LN,.
   IF IBAC NE 1
   THEN GO TO RETURN,.                /**********************************/
BACK..                                /*BACKSUBSTITUTION             */
        DO I =LN TO 1 BY -1,.         /**********************************/
        PIV  =A(I,1),.
        IF PIV= 0                     /*TEST FOR ZERO PIVOT          */
        THEN DO,.
             ERROR='S',.              /*PIVOT ELEMENT IS ZERO        */
             GO TO RETURN,.
             END,.
        INL  =I-1,.
             DO J =1 TO LM,.          /*LOOP OVER ALL COLUMNS OF R   */
             SUM  =R(I,J),.
                  DO K =2 TO IBAC,.   /*CALCULATE SCALAR PRODUCT     */
                  SUM  =SUM-MULTIPLY(A(I,K),R(INL+K,J),53),.
                  END,.
             R(I,J)=SUM/PIV,.         /*COMPUTE NEW ELEMENT IN R     */
             END,.
        IF IBAC LT NB
        THEN IBAC =IBAC+1,.           /*UPDATE END OF INNERMOST LOOP */
        END,.
RETURN..
   END,.                              /*END OF PROCEDURE MGB         */
```

Purpose:

MGB1 performs the following operations on an equation system A · X = R with general band matrix A = L · U, depending on the character of an input parameter OPT:

OPT = 'L'	U replaces A and $L^{-1}R$ replaces R
OPT = 'U'	U replaces A and $U^{-1}R$ replaces R
OPT = 'B'	$U^{-1}R$ replaces R for a given U on storage locations of A
otherwise	U replaces A and the solution $X = A^{-1}R$ replaces R

The following table shows input and output depending on OPT:

MGB1 - OPT	'L'		'U'		'B'		otherwise	
INPUT	A	R	A	R	U	R	A	R
OUTPUT	U	$L^{-1}\cdot R$	U	$U^{-1}\cdot R$	U	$U^{-1}\cdot R$	U	$A^{-1}\cdot R$

Usage:

CALL MGB1 (A, R, N, NLD, NUD, M, EPS, OPT);

A(N, NB) - BINARY FLOAT [(53)]
Given N by N band matrix A consisting of the main diagonal, NLD lower codiagonals, and NUD upper codiagonals. A is stored rowwise and left-adjusted so that A(i, 1) contains the first nontrivial element in the i-th row of matrix A, i=1, 2, ..., N. Thus, the maximum number of elements in the rows of array A is:
NB = min (N, NLD + NUD + 1)
Resultant upper band factor U stored rowwise and left-adjusted so that A(i, 1) contains the diagonal element in the i-th row of the upper factor U, i=1, 2, ..., N. If OPT = 'B', A contains U.

R(N, M) - BINARY FLOAT [(53)]
Given right-hand-side matrix with N rows and M columns, which implies that M sets of right-hand-side vectors are given.
Resultant solution depending on the option parameter OPT (see "Purpose").

N - BINARY FIXED
Given row dimension of matrix A and number of rows of right-hand side R.

NLD - BINARY FIXED
Given number of lower codiagonals of matrix A.

NUD - BINARY FIXED
Given number of upper codiagonals of matrix A.

M - BINARY FIXED
Given number of columns of R, that is, number of right-hand-side vectors.

EPS - BINARY FLOAT
Given relative tolerance for test on loss of significant digits.

OPT - CHARACTER(1)
Given option parameter for selection of operation (see "Purpose").

Purpose:

MGB2 performs the following operations on an equation system A · X = R with general band matrix A = L · U, depending on the character of an input parameter OPT:

OPT = 'L' A is replaced by upper band factor U, R is replaced by $L^{-1}\cdot R$, and lower band factor L is stored in a one-dimensional array L omitting the unit diagonal.

OPT = 'F' A is replaced by the upper band factor U and the lower band factor L is stored in the array L. The right-hand side R remains unchanged.

OPT = 'A' R is replaced by $L^{-1}\cdot R$ for the given upper factor U in array A and the lower factor L in vector L.

OPT = 'C' R is replaced by the solution $X = A^{-1}\cdot R$ for given U and L.

otherwise A is replaced by the upper factor U. The lower factor L is calculated and stored in L, and R is replaced by the solution $X = A^{-1}\cdot R$.

The following table shows input and output depending on OPT:

MGB2 - OPT	'L'			'F'			'A'			'C'			otherwise		
INPUT	A		R	A		R	U	L	R	U	L	R	A		R
OUTPUT	U	L	$L^{-1}\cdot R$	U	L	R	U	L	$L^{-1}\cdot R$	U	L	$A^{-1}\cdot R$	U	L	$A^{-1}\cdot R$

Usage:

CALL MGB2 (A, R, L, IPER, N, NLD, NUD, M, EPS, OPT);

A(N, NB) - BINARY FLOAT [(53)]
Given an N by N band matrix A consisting of the main diagonal, NLD lower codiagonals, and NUD upper codiagonals. A is stored rowwise and left-adjusted so that A(i, 1) contains the first nontrivial element in the i-th row of matrix A. Thus, the maximum number of elements in the rows of the array A is:
NB = min (N, NLD + NUD + 1)
Resultant upper band factor U stored rowwise and left-adjusted so that A(i, 1) contains the diagonal element in i-th row of U, i = 1, 2, ..., N.
If OPT = 'A' or 'C', the array A contains U.

R(N, M) - BINARY FLOAT [(53)]
Given right-hand-side matrix with N rows and M columns, which implies that M sets of right-hand-side vectors are given.
Resultant solution depending on the option parameter OPT (see "Purpose").

L(N· NLD-NLD· (NLD+1)/2)
BINARY FLOAT [(53)]
Resultant one-dimensional array containing the lower factor L. If OPT = 'A' or 'C', array L contains the lower factor L, obtained by subroutine MGB2 with any other option parameter.

IPER(N) - BINARY FIXED
Resultant integer vector containing the permutations of rows of the matrix A in the factorization steps.
If OPT = 'A' or 'C', permutation vector IPER must be given, obtained by MGB2 with OPT = 'A', 'C'.

N - BINARY FIXED
Given row dimension of matrix A and number of rows of right-hand side R.

NLD - BINARY FIXED
Given number of lower codiagonals of the matrix A.

NUD - BINARY FIXED
Given number of upper codiagonals of the matrix A.

M - BINARY FIXED
Given number of columns of R, that is, number of right-hand-side vectors.

EPS - BINARY FLOAT
Given relative tolerance for test on loss of significant digits.

OPT - CHARACTER(1)
Given option parameter for selection of operation (see "Purpose").

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='P' means error in specified parameters:
$M \le 0$ or $NLD < 0$ or $N \le NLD$
or $NUD < 0$ or $N \le NUD$

ERROR='S' means all elements in a row of the given matrix A are zero, or the calculated pivot in a factorization step is zero. This is possibly due to an ill-conditioned or singular matrix A.

ERROR='W' is a warning indicating possible loss of significance.

The storage mode for band matrices is a natural generalization of the normal two-dimensional storage scheme: any row is stored with NB=min (N, NLD+1+NUD) elements, but only the nontrivial elements (that is, those within the band) must be specified. The remaining elements are set to zero automatically within procedure MGB1/MGB2.

Note that a fully populated N by N matrix would require exactly N • N storage locations if stored as band matrix in compressed form. However, the unit lower triangular factor L would need additional N • (N-1)/2 storage locations.

Method:

Calculations of the lower and upper band factors L, U are done using a standard Gaussian elimination technique. Columnwise pivoting is built in, combined with scaling of rows (equilibration).

The lower band factor L is normalized such that the diagonal contains all ones, which are not stored (Doolittle factorization).

The procedure gets the required solutions by means of forward and/or backward substitutions, where the interchange information is combined with the lower band factor L.

For reference see:

R.S. Martin and J.H. Wilkinson, "Solution of Symmetric and Unsymmetric Band Equations on the

Calculation of Eigenvectors of Band Matrices", Numerische Mathematik, vol. 9, 1967, pp. 279-301.

Mathematical Background:

Let A be an N by N nonsingular real band matrix with NLD lower codiagonals and NUD upper codiagonals. In general, it can be factorized into a product

$$A = P \cdot L \cdot U$$

where L and U are lower and upper band factors respectively. L can be normalized so that it has a unit diagonal. P means the row-permutation matrix, that is, an N by N unit matrix with interchanged rows resulting from partial pivoting in the factorization steps.

Then $X = L^{-1} \cdot P^{-1} \cdot R = L^{-1}\,\overline{R}$ is calculated using forward substitution to obtain X from $L \cdot X = P^{-1} \cdot R = \overline{R}$, where $\overline{R}$ is obtained from R by interchanging rows in the same way that rows of matrix A are interchanged during columnwise pivoting in factorization.

Calculation of $Y = U^{-1} \cdot R$ is done using backward substitution to obtain Y from $U \cdot Y = R$.

Calculation of $Z = U^{-1} \cdot L^{-1} \cdot P^{-1} \cdot R = U^{-1} \cdot L^{-1} \cdot \overline{R}$ is done by first solving $L \cdot X = \overline{R}$ and then solving $U \cdot Z = X$.

Programming Considerations:

1. Storage Mode

The following is an example of a 7 by 7 matrix with two lower and three upper codiagonals which shows the storage compression of band matrices and the storage allocation of upper and lower triangular factors U and L.

Fully stored matrix:

$$\begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & 0 & 0 & 0 \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 & 0 \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & 0 \\
0 & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\
0 & 0 & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \\
0 & 0 & 0 & a_{61} & a_{62} & a_{63} & a_{64} \\
0 & 0 & 0 & 0 & a_{71} & a_{72} & a_{73}
\end{pmatrix}$$

Compressed stored band matrix:

$$\begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & x & x \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & x \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & x \\
a_{61} & a_{62} & a_{63} & a_{64} & x & x \\
a_{71} & a_{72} & a_{73} & x & x & x
\end{pmatrix}$$

Elements marked X need not be specified. They get filled up with zeros automatically.

Resultant upper triangular factor U and unit lower triangular factor L:

$$U = \begin{pmatrix}
u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & u_{16} \\
u_{22} & u_{23} & u_{24} & u_{25} & u_{26} & u_{27} \\
u_{33} & u_{34} & u_{35} & u_{36} & u_{37} & 0 \\
u_{44} & u_{45} & u_{46} & u_{47} & 0 & 0 \\
u_{55} & u_{56} & u_{57} & 0 & 0 & 0 \\
u_{66} & u_{67} & 0 & 0 & 0 & 0 \\
u_{77} & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$L = \begin{pmatrix}
l_{21} & & & & & \\
l_{31} & l_{32} & & & & \\
 & l_{42} & l_{43} & & & \\
 & & l_{53} & l_{54} & & \\
 & & & l_{64} & l_{65} & \\
 & & & & l_{75} & l_{76}
\end{pmatrix}$$

The band-shaped upper triangular factor U is stored rowwise and left-adjusted, so that A(i, 1) contains the diagonal element for i = 1, 2, ..., N. The band-shaped lower unit triangular factor L is stored in a one-dimensional array. Only the nontrivial subdiagonal elements are stored columnwise in successive storage locations.

2. Computational remarks

In order to improve numerical stability, partial pivoting is used combined with an equilibration of rows. In each row i of the given matrix A the element a_{ij_i} of greatest absolute value is found. The absolute values $v_i = 1/\left|a_{ij_i}\right|$ are used as weights for pivoting:

At the first step of Gaussian elimination that element a_{kl} is used as pivot element piv for which

$$\left|a_{k1}\right| \cdot v_k = \max_{i=1,\dots,NLD+1} \left(\left|a_{i1}\right| \cdot v_i\right)$$

If necessary, rows k and l are interchanged in A, R and $V = \begin{pmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ v_N \end{pmatrix}$ and IPER(1) is set to k.

The elements in the first NLD rows are transformed by means of

$$l_{i1} = \frac{a_{i1}}{piv} \qquad i = 2,\dots,NLD+1$$

$$a_{ij}^{(1)} = a_{ij} - l_{i1} \cdot a_{1j} \qquad j = 2,\dots,NB$$

$$r_{ik}^{(1)} = r_{ik} - l_{i1} \cdot r_{1k} \qquad k = 1,\dots,M$$

If specified, the elements l_{i1} are stored in successive locations within L.

Transformed rows of A get shifted to the left by one position, and zero is inserted in the last location.

Repeating this process (N-1) times gives triangular factors U and L and the product $L^{-1}R$, in permuted form.

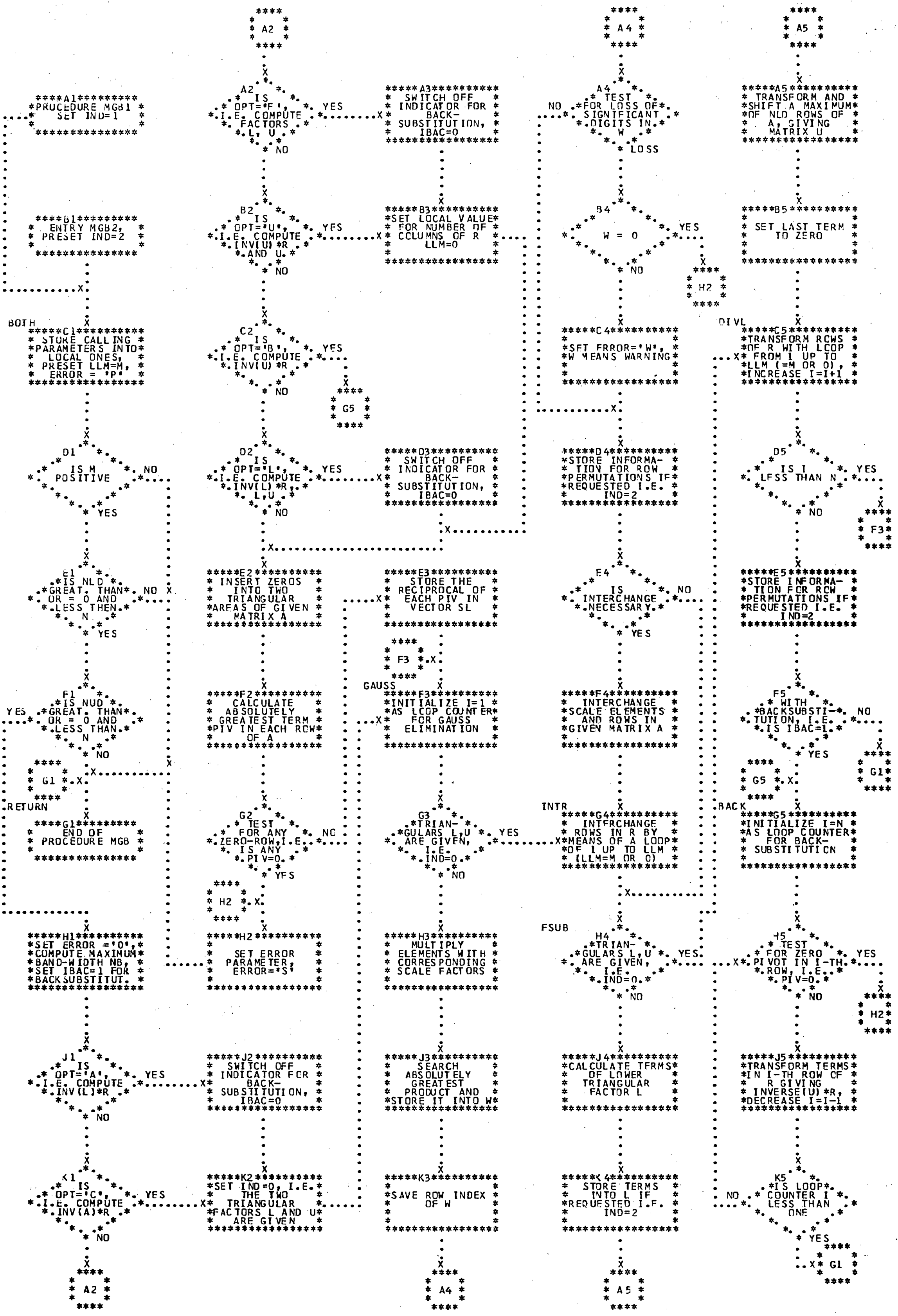
If at an elimination step the value of piv becomes zero, then ERROR is set to 'S' and further calculation is bypassed.

ERROR is set to 'W' if, at elimination step j, $v_j \cdot piv \leq EPS$.

FOR AN EQLATION SYSTEM A*X=R WITH BAND MATRIX A=L*U

MGB1 CPTIONALLY COPPUTES X, U, INVERSE(L)*R, INVERSE(U)*R

MGB2 CPTIONALLY COPPUTES X, U, INVERSE(L)*R, AND L



A1 PROCEDURE MGB1 SET IND=1
B1 ENTRY MGB2, PRESET IND=2
BOTH
C1 STORE CALLING PARAMETERS INTO LOCAL ONES, PRESET LLM=M, ERROR = 'P'
D1 IS M POSITIVE
NO
YES
E1 IS NLD GREAT. THAN OR = 0 AND LESS THEN N
NO
YES
F1 IS NUD GREAT. THAN OR = 0 AND LESS THAN N
YES
NO
G1
RETURN
G1 END OF PROCEDURE MGB
H1 SET ERROR ='0', COMPUTE MAXIMUM BAND-WIDTH NB, SET IBAC=1 FOR BACKSUBSTITUT.
J1 IS OPT='A', I.E. COMPUTE INV(L)*R
YES
NO
K1 IS OPT='C', I.E. COMPUTE INV(A)*R
YES
NO
A2
A2 IS OPT='F', I.E. COMPUTE FACTORS L, U
YES
NO
B2 IS OPT='U', I.E. COMPUTE INV(U)*R AND U
YES
NO
C2 IS OPT='B', I.E. COMPUTE INV(U)*R
YES
NO
G5
D2 IS OPT='L', I.E. COMPUTE INV(L)*R, L,U
YES
NO
E2 INSERT ZEROS INTO TWO TRIANGULAR AREAS OF GIVEN MATRIX A
F2 CALCULATE ABSOLUTELY GREATEST TERM PIV IN EACH ROW OF A
G2 TEST FOR ANY ZERO-ROW, I.E. IS ANY PIV=0.
NO
YES
H2
H2 SET ERROR PARAMETER, ERROR='S'
J2 SWITCH OFF INDICATOR FOR BACK-SUBSTITUTION, IBAC=0
K2 SET IND=0, I.E. THE TWO TRIANGULAR FACTORS L AND U ARE GIVEN
A2
A3 SWITCH OFF INDICATOR FOR BACK-SUBSTITUTION, IBAC=0
B3 SET LOCAL VALUE FOR NUMBER OF COLUMNS OF R LLM=0
D3 SWITCH OFF INDICATOR FOR BACK-SUBSTITUTION, IBAC=0
E3 STORE THE RECIPROCAL OF EACH PIV IN VECTOR SL
F3
GAUSS
F3 INITIALIZE I=1 AS LOOP COUNTER FOR GAUSS ELIMINATION
G3 TRIAN-GULARS L,U ARE GIVEN, I.E. IND=0
YES
NO
H3 MULTIPLY ELEMENTS WITH CORRESPONDING SCALE FACTORS
J3 SEARCH ABSOLUTELY GREATEST PRODUCT AND STORE IT INTO W
K3 SAVE ROW INDEX OF W
A4
A4
A4 TEST FOR LOSS OF SIGNIFICANT DIGITS IN W
NO
LOSS
B4 W = 0
YES
NO
H2
C4 SET ERROR='W', W MEANS WARNING
D4 STORE INFORMATION FOR ROW PERMUTATIONS IF REQUESTED I.E. IND=2
E4 IS INTERCHANGE NECESSARY
NO
YES
F4 INTERCHANGE SCALE ELEMENTS AND ROWS IN GIVEN MATRIX A
INTR
G4 INTERCHANGE ROWS IN R BY MEANS OF A LOOP OF 1 UP TO LLM (LLM=M OR 0)
FSUB
H4 TRIAN-GULARS L,U ARE GIVEN, I.E. IND=0
YES
NO
J4 CALCULATE TERMS OF LOWER TRIANGULAR FACTOR L
K4 STORE TERMS INTO L IF REQUESTED I.E. IND=2
A5
A5
A5 TRANSFORM AND SHIFT A MAXIMUM OF NLD ROWS OF A, GIVING MATRIX U
B5 SET LAST TERM TO ZERO
DIVL
C5 TRANSFORM ROWS OF R WITH LOOP FROM 1 UP TO LLM (=M OR 0), INCREASE I=I+1
D5 IS I LESS THAN N
YES
NO
F3
E5 STORE INFORMATION FOR ROW PERMUTATIONS IF REQUESTED I.E. IND=2
F5 WITH BACKSUBSTITUTION, I.E. IS IBAC=1
NO
YES
G1
G5
BACK
G5 INITIALIZE I=N AS LOOP COUNTER FOR BACK-SUBSTITUTION
H5 TEST FOR ZERO PIVOT IN I-TH. ROW, I.E. PIV=0.
YES
NO
H2
J5 TRANSFORM TERMS IN I-TH ROW OF R GIVING INVERSE(U)*R, DECREASE I=I-1
K5 IS LOOP COUNTER I LESS THAN ONE
NO
YES
G1

Eigenvalues and Related Topics

Note: The following example illustrates a way to link subroutines MATE, MEAT, MVAT, MVEB (which follow) for the computation of the eigenvalues and eigenvectors of a real nonsymmetric matrix. (Subroutines MATE and MVEB can be replaced with MATU and MVUB.)

Description of the parameters used:

A - Real array containing the given matrix (this matrix is not preserved)

N - Order of the matrix

H - Real array in which the Hessenberg matrix will be saved together with the elements of the transformations involved in subroutine MATE

CH - Complex array containing the Hessenberg matrix for the computation of the eigenvectors

EV - Complex array where the eigenvectors are stored

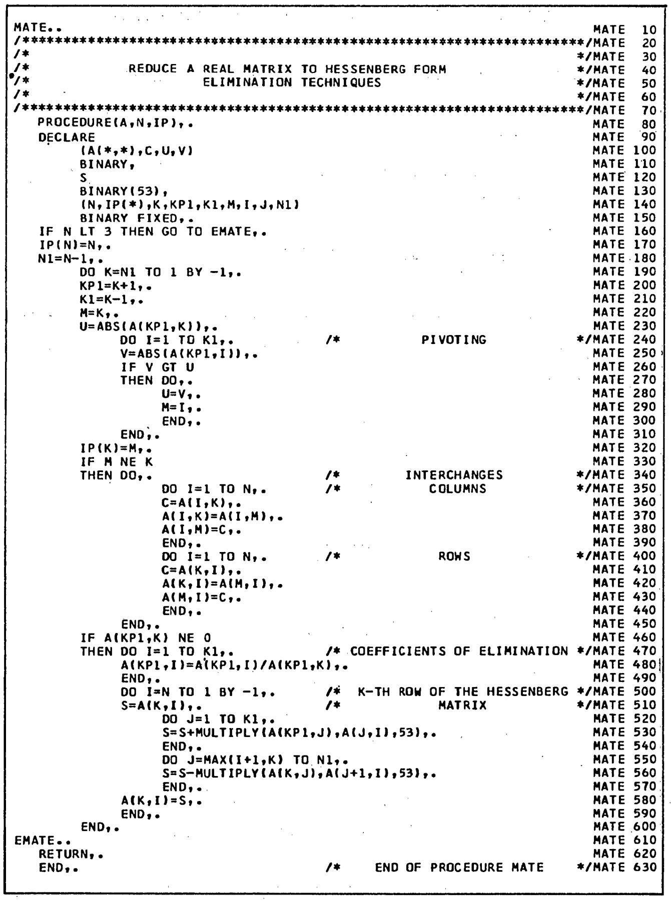
The other parameters are defined in the descriptions of the subroutines.

All the eigenvalues are assumed to be complex in this example, so that only N/2 eigenvectors are computed.

```
. . . . . .                                   / *          MAIN PROGRAM           * /
   N = 50, .
        BEGIN, .                              / *          BEGIN BLOCK            * /
        DECLARE
             (A(N,N),RR(N),RI(N),H(N,N))            BINARY,
             (CH(N,N),EIG,EV(N,N/2))                COMPLEX BINARY,
             (IP(N),I,J,K,M)                        BINARY FIXED,
             ANA(N)                                 BIT(1), .
        CALL  GEN(A,N), .                     / *       GENERATE THE MATRIX        * /
        CALL  MATE(A,N,IP), .                 / *  REDUCTION TO HESSENBERG FORM    * /
        H = A, .                              / *     SAVE HESSENBERG MATRIX       * /
        CALL  MEAT(A,N,RR,RI,ANA), .          / *    COMPUTE THE EIGENVALUES       * /
        I = 0, .
             DO M = 1 TO N BY 2, .            / *    COMPUTE N/2 EIGENVECTORS      * /
             I = I + 1, .
             EIG=COMPLEX(RR(M),RI(M)), .
             CH(1,*) = H (1,*), .             / *    PUT THE HESSENBERG MATRIX     * /
                 DO J = 2 TO N, .             / *      INTO A COMPLEX ARRAY        * /
                      DO K=J-1 TO N, .
                      CH(J,K)=H (J,K), .
                      END, .
                 END, .
             CALL MVAT (CH,N,EIG,EV(*,I)), . / * EIGENVECTORS OF THE               * /
                                              / *    HESSENBERG MATRIX             * /
             CALL MVEB(H,N,IP,EV(*,I)), . / * VECTORS OF THE GIVEN MATRIX          * /
             END, .
        PUT  EDIT . . . . .                   / *    PRINT THE RESULTS             * /
        END, .                                / *     END BEGIN BLOCK              * /
 . . . . . .                                  / *      MAIN PROGRAM
```

Note that the eigenvalues of the original matrix A are equal to the eigenvalues of the corresponding Hessenberg matrix, so that no back transformation of the eigenvalues is required.

- Subroutine MATE



```
MATE..
/************************************************************************/
/*                                                                      */
/*           REDUCE A REAL MATRIX TO HESSENBERG FORM                    */
/*                  ELIMINATION TECHNIQUES                              */
/*                                                                      */
/************************************************************************/
   PROCEDURE(A,N,IP),.
   DECLARE
        (A(*,*),C,U,V)
        BINARY,
        S
        BINARY(53),
        (N,IP(*),K,KP1,K1,M,I,J,N1)
        BINARY FIXED,.
   IF N LT 3 THEN GO TO EMATE,.
   IP(N)=N,.
   N1=N-1,.
        DO K=N1 TO 1 BY -1,.
        KP1=K+1,.
        K1=K-1,.
        M=K,.
        U=ABS(A(KP1,K)),.
             DO I=1 TO K1,.         /*          PIVOTING                  */
             V=ABS(A(KP1,I)),.
             IF V GT U
             THEN DO,.
                  U=V,.
                  M=I,.
                  END,.
             END,.
        IP(K)=M,.
        IF M NE K
        THEN DO,.                   /*        INTERCHANGES                */
                  DO I=1 TO N,.     /*          COLUMNS                   */
                  C=A(I,K),.
                  A(I,K)=A(I,M),.
                  A(I,M)=C,.
                  END,.
                  DO I=1 TO N,.     /*            ROWS                    */
                  C=A(K,I),.
                  A(K,I)=A(M,I),.
                  A(M,I)=C,.
                  END,.
             END,.
        IF A(KP1,K) NE 0
        THEN DO I=1 TO K1,.         /* COEFFICIENTS OF ELIMINATION */
             A(KP1,I)=A(KP1,I)/A(KP1,K),.
             END,.
             DO I=N TO 1 BY -1,.    /*  K-TH ROW OF THE HESSENBERG */
             S=A(K,I),.             /*            MATRIX                  */
                  DO J=1 TO K1,.
                  S=S+MULTIPLY(A(KP1,J),A(J,I),53),.
                  END,.
                  DO J=MAX(I+1,K) TO N1,.
                  S=S-MULTIPLY(A(K,J),A(J+1,I),53),.
                  END,.
             A(K,I)=S,.
             END,.
        END,.
EMATE..
   RETURN,.
   END,.                            /*    END OF PROCEDURE MATE     */
```

Purpose:

MATE reduces a given real matrix to upper almost triangular (Hessenberg) form by means of a sequence of similarities.

Usage:

CALL MATE (A, N, IP);

A(N, N) - BINARY FLOAT
Given real matrix.
Resultant upper almost triangular matrix.

N - BINARY FIXED
Given order of the matrix.

IP(N) - BINARY FIXED
Resultant vector containing information about the interchanges operated on rows and columns of the matrix.

Remarks:

The elements defining the transformations applied to the matrix are stored in place of the lower triangular part of the matrix on return. These elements and the vector IP will be used in the computation of the eigenvectors of the original matrix (Procedure MVEB).

Method:

Each row of the matrix is reduced in turn, starting from the last one, by applying a suitable elimination, and similarity is achieved by applying the left inverse transformation. A Crout-like algorithm is used to take advantage of the accumulation of the inner products in double precision.

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical Background:

Let us consider a matrix A of order n and the similarity

$$TAT^{-1} = H \qquad (1)$$

where H is a Hessenberg matrix associated with A, and T a lower triangular matrix with unit diagonal. Equation (1) can be written as

$$TA = HT \qquad (2)$$

Matrices H and T will be determined row by row, according to the algorithm described below.

If rows (k+1) to n of H and rows k to n of T are assumed to be known, row k of H and row (k-1) of T will be determined as follows.

From equation (2) we get

$$a_{ki} = \sum_{j=1}^{k-1} t_{kj}\, a_{ji} = h_{ki} + \sum_{j=i+1}^{n} h_{kj}\, t_{ji}$$

and

$$h_{ki} = a_{ki} + \sum_{j=1}^{k-1} t_{kj}\, a_{ji} - \sum_{j=i+1}^{n} h_{kj}\, t_{ji} \qquad (3)$$

If we apply equation (3) for $i = n, n-1, \ldots, k$, we will obtain recursively the terms of the k-th row of H, excepting the subdiagonal term. (When the upper bound of a summation is less than the lower bound, the value of the sum is taken as zero.)

Let us determine now the (k-1)st row of T and the subdiagonal term

$h_{k\,k-1}$ of H.

From equation (2) we get

$$a_{ki} + \sum_{j=1}^{k-1} t_{kj}\, a_{ji} = \sum_{j=k-1}^{n} h_{kj}\, t_{ji}, \quad 1 \le i \le k-1$$

Defining

$$m_{ki} = a_{ki} + \sum_{j=1}^{k-1} t_{kj}\, a_{ji} - \sum_{j=k}^{n} h_{kj}\, t_{ji}, \quad 1 \le i \le k-1 \qquad (4)$$

we finally obtain

$$h_{k\,k-1} = m_{k\,k-1}, \quad t_{k-1\,i} = \frac{m_{ki}}{h_{k\,k-1}}, \quad 1 \le i \le k-2 \qquad (5)$$

To ensure stability, a technique of pivoting is incorporated in this algorithm.

After the computation of the m_{ki}'s, the subscript j is determined for which

$$\left| m_{kj} \right| \ge \left| m_{ki} \right|, \quad 1 \le i \le k-1.$$

Then the elements m_{kj} and $m_{k\,k-1}$ are interchanged. So are columns j and (k-1) of T. Similarly, the columns and the rows of matrix A are also interchanged. Then equations (4) and (5) are applied.

The algorithm is initialized by taking

$$h_{nn} = a_{nn}$$

$$\left.\begin{array}{l} m_{ni} = a_{ni} \\ t_{ni} = 0 \end{array}\right\} \quad 1 \le i \le n-1$$

$$t_{nn} = 1$$

When $m_{ki} = 0$ for $i = 1, \ldots, k-1$, $h_{k\,k-1} = 0$ and $t_{k-1\,i} = 0$ for $i = 1, \ldots, k-2$.

Programming considerations:

1. The interchanges determined by the pivoting are stored in vector IP. This vector will be used in the computation of the eigenvectors (subroutine MVEB).

2. The matrix T is stored in the lower part of the array A, overwriting the terms of the original matrix:

$$t_{I,J} \rightarrow A\,(I+1,\ J), \qquad 2 \le I \le N-1,\ 1 \le J \le I-1$$

These elements $t_{I,J}$ will be used in the computation of the eigenvectors (subroutine MVEB). The last row and the diagonal of T are not stored.

3. The inner products involved in equations (3) and (4) are computed in double precision.

- <u>Subroutine</u> MATU

```
MATU..
/***********************************************************************/
/*                                                                     */
/*            REDUCE A REAL MATRIX TO HESSENBERG FORM                  */
/*                  HOUSEHOLDER'S TRANSFORMATIONS                      */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (A,N,B),.
   DECLARE
        (A(*,*),B(*),EPS,T,C,U) BINARY,
        S BINARY(53),
        (I,J,K,KP1,KP2,N) BINARY FIXED,.
   EPS=1.0E-14,.
   B(1) =0,.
        DO K=1 TO N-2,.
        KP1  =K+1,.
        KP2  =KP1+1,.
        S    =0,.                    /* PREPARE K-TH TRANSFORMATION */
             DO I=KP2 TO N,.
             S=S+MULTIPLY(A(I,K),A(I,K),53),.
             END,.
        T    =A(KP1,K)*A(KP1,K),.
        IF S GT EPS*T
        THEN DO,.
        S    =SQRT(S+T),.
             T    =S,.               /*  CHOOSE SIGN FOR STABILITY  */
             IF A(KP1,K) GT 0 THEN T=-T,.
             C    =A(KP1,K)-T,.
                  DO J=KP1 TO N,.    /*         ROW OPERATION       */
                  S    =0,.
                       DO I=KP1 TO N,.
                       S=S+MULTIPLY(A(I,J),A(I,K),53),.
                       END,.
                  U    =A(KP1,J),.
                  A(KP1,J)=S/T,.
                  U    =(A(KP1,J)-U)/C,.
                       DO I=KP2 TO N,.
                       A(I,J)=A(I,J)+U*A(I,K),.
                       END,.
                  END,.
                  DO J=1 TO N,.      /*       COLUMN OPERATION      */
                  S    =0,.
                       DO I=KP1 TO N,.
                       S=S+MULTIPLY(A(J,I),A(I,K),53),.
                       END,.
                  U    =A(J,KP1),.
                  A(J,KP1)=S/T,.
                  U    =(A(J,KP1)-U)/C,.
                       DO I=KP2 TO N,.
                       A(J,I)=A(J,I)+U*A(I,K),.
                       END,.
                  END,.
             B(KP1)=A(KP1,K),.
             A(KP1,K)=T,.            /*  TRANSFORM SUBDIAGONAL TERM */
             END,.
        ELSE B(KP1)=0,.              /*  BYPASS K-TH TRANSFORMATION */
        END,.
   RETURN,.
   END,.                             /*    END OF PROCEDURE MATU    */
```

Purpose:

MATU reduces a given real matrix to upper almost triangular (Hessenberg) form by means of a sequence of orthogonal transformations.

Usage:

CALL MATU (A, N, B);

A(N, N) - BINARY FLOAT
Given real matrix.
Resultant upper almost triangular matrix.

N - BINARY FIXED
Given order of the matrix.

B(N) - BINARY FLOAT
Resultant vector containing information about the transformations applied to the original matrix.

Remarks:

Other elements defining the transformations are stored in place of the lower triangular part of the matrix on return. These elements and the vector B will be used in the computation of the eigenvectors of the original matrix (Procedure MVUB).

Method:

Each column of the matrix is reduced in turn by means of orthogonal similarities (Householder's transformations).

For reference see:

J. H. Wilkinson, <u>The Algebraic Eigenvalue Problem</u>, Clarendon Press, Oxford, 1965.

Mathematical Background:

For a given real matrix A of order n, let us consider the sequence of similarities

$$A^{(i+1)} = P_i A^{(i)} P_i^{-1} \quad i = 1, 2, \dots, n-2$$

$$\text{with } A^{(1)} = A \qquad (1)$$

Assuming that $A^{(i)}$ is of almost triangular form in its first (i-1) columns, we will determine a transformation P_i such that $A^{(i+1)}$ is of almost triangular form in its first i columns. Among the matrices P_i, let us consider those of the form

$$P_i = I - 2 u u^T \quad \text{(Householder's matrices)} \qquad (2)$$

where I is the unit matrix and u a vector of order n such that

$$< u, u > = 1 \qquad (3)$$

These matrices are orthogonal and symmetric, and equation (1) can be written as

$$A^{(i+1)} = P_i A^{(i)} P_i \qquad (4)$$

Let us now define a vector v by

$$v^T = (v_1, v_2, \dots, v_n),$$

with

$$v_k = 0 \text{ for } k = 1, 2, \dots, i$$

$$v_k = a_{k,i}^{(i)} \text{ for } k = i + 1, \dots, n$$

and try to determine the transformation P_i so that

$$P_i v = be_{i+1} \quad \text{where } b = \pm < v, v>^{1/2} \qquad (5)$$

e_{i+1} is a vector whose components are zero, except for the (i+1) st which is one.

The combination of equations (2) and (5) gives

$$P_i v = v - 2 < u, v > u$$

$$= be_{i+1}$$

Putting $< u, v > = s$, u is given by

$$u = \frac{v - be_{i+1}}{2s}$$

From equation (3) we get

$$s^2 = b (b - v_{i+1})/2$$

Then the matrix P_i can be written as

$$P_i = I + \frac{1}{b(v_{i+1} - b)} (v - be_{i+1}) (v - be_{i+1})^T$$

The sign of b will be such that the magnitude of the denominator is maximum, that is,

$$\text{sign } (b) = - \text{ sign } (v_{i+1})$$

in order to ensure stability.

If we now form the product $P_i A^{(i)}$, the resulting matrix, according to (5), will have zeros in positions (k, i), k = 1 + 2, . . . , n, and the term in position (i+1, i) will be b. The (i-1) first columns and rows remain unaltered.

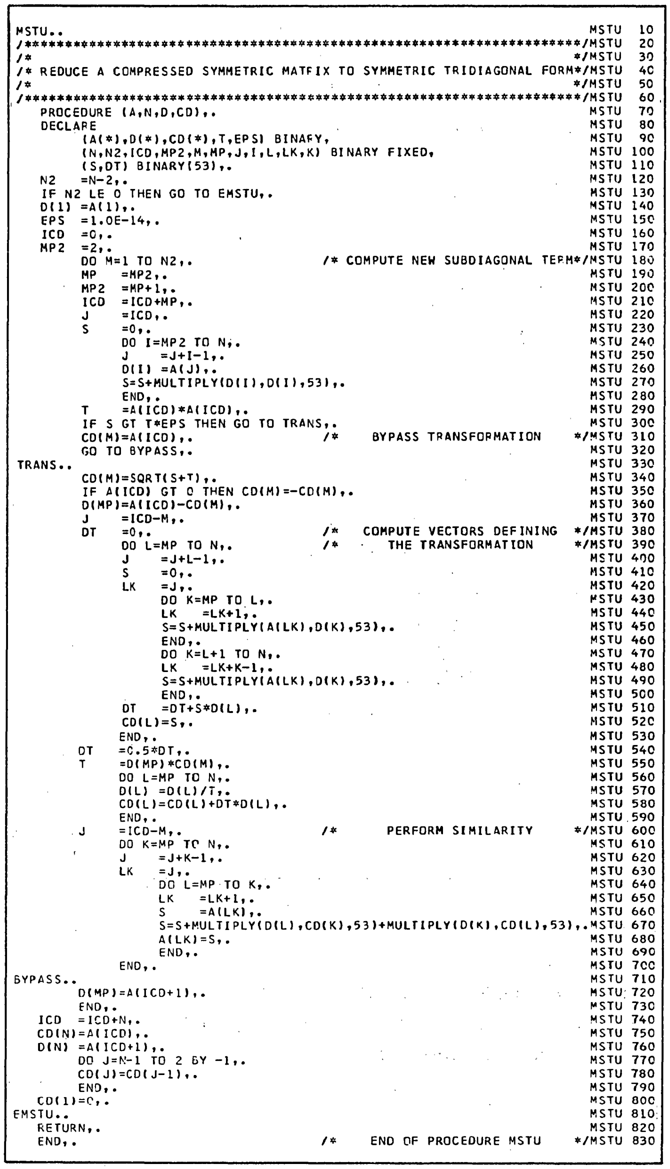
The right transformation $(P_i A^{(i)}) P_i$, completing the similarity, will leave this structure unchanged. Thus, after (n-2) transformations according to (1) and (4), the matrix will be reduced to almost triangular form.

When the matrix is symmetric, it is interesting to note that the resulting almost triangular form is symmetric also (that is, tridiagonal).

Programming Considerations:

A transformation P_i for which $| v_{i+1} + b | < 10^{-7} | b |$ is bypassed. All the scalar products involved in the computation are calculated in double precision.

• Subroutine MSTU



```
MSTU..
/**********************************************************************/
/*                                                                    */
/* REDUCE A COMPRESSED SYMMETRIC MATFIX TO SYMMETRIC TRIDIAGONAL FORM*/
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,N,D,CD),.
   DECLARE
        (A(*),D(*),CD(*),T,EPS) BINAFY,
        (N,N2,ICD,MP2,M,MP,J,I,L,LK,K) BINARY FIXED,
        (S,DT) BINARY(53),.
   N2   =N-2,.
   IF N2 LE 0 THEN GO TO EMSTU,.
   D(1) =A(1),.
   EPS  =1.0E-14,.
   ICD  =0,.
   MP2  =2,.
        DO M=1 TO N2,.                 /* COMPUTE NEW SUBDIAGONAL TEFM*/
        MP   =MP2,.
        MP2  =MP+1,.
        ICD  =ICD+MP,.
        J    =ICD,.
        S    =0,.
             DO I=MP2 TO N,.
             J    =J+I-1,.
             D(I) =A(J),.
             S=S+MULTIPLY(D(I),D(I),53),.
             END,.
        T    =A(ICD)*A(ICD),.
        IF S GT T*EPS THEN GO TO TRANS,.
        CD(M)=A(ICD),.                 /*    BYPASS TRANSFORMATION    */
        GO TO BYPASS,.
TRANS..
        CD(M)=SQRT(S+T),.
        IF A(ICD) GT 0 THEN CD(M)=-CD(M),.
        D(MP)=A(ICD)-CD(M),.
        J    =ICD-M,.
        DT   =0,.                      /*   COMPUTE VECTORS DEFINING  */
             DO L=MP TO N,.            /*     THE TRANSFORMATION      */
             J    =J+L-1,.
             S    =0,.
             LK   =J,.
                  DO K=MP TO L,.
                  LK   =LK+1,.
                  S=S+MULTIPLY(A(LK),D(K),53),.
                  END,.
                  DO K=L+1 TO N,.
                  LK   =LK+K-1,.
                  S=S+MULTIPLY(A(LK),D(K),53),.
                  END,.
             DT   =DT+S*D(L),.
             CD(L)=S,.
             END,.
        DT   =C.5*DT,.
        T    =D(MP)*CD(M),.
             DO L=MP TO N,.
             D(L) =D(L)/T,.
             CD(L)=CD(L)+DT*D(L),.
             END,.
        J    =ICD-M,.                  /*      PERFORM SIMILARITY     */
             DO K=MP TC N,.
             J    =J+K-1,.
             LK   =J,.
                  DO L=MP TO K,.
                  LK   =LK+1,.
                  S    =A(LK),.
                  S=S+MULTIPLY(D(L),CD(K),53)+MULTIPLY(D(K),CD(L),53),.
                  A(LK)=S,.
                  END,.
             END,.
BYPASS..
        D(MP)=A(ICD+1),.
        END,.
   ICD  =ICD+N,.
   CD(N)=A(ICD),.
   D(N) =A(ICD+1),.
        DO J=N-1 TO 2 BY -1,.
        CD(J)=CD(J-1),.
        END,.
   CD(1)=C,.
EMSTU..
   RETURN,.
   END,.                               /*    END OF PROCEDURE MSTU    */
```

Purpose:

MSTU reduces a given real symmetric matrix to tridiagonal form by means of a sequence of orthogonal transformations.

Usage:

CALL MSTU (A, N, D, CD);

A(N*(N+1)/2) -	BINARY FLOAT Given matrix in compressed storage mode.
N -	BINARY FIXED Given order of the matrix.

D(N) - BINARY FLOAT
Resultant vector containing the diagonal terms of the tridiagonal matrix.

CD(N) - BINARY FLOAT
Resultant vector containing the codiagonal terms of the tridiagonal matrix in positions 2, 3, ..., N.

Remarks:

The elements defining the transformations applied to the matrix will replace the given matrix in array A. These elements will be used in the computation of the eigenvectors of the original matrix (subroutine MVSU).

Method:

Each row and column of the matrix is reduced in turn by means of orthogonal similarities (Householder's transformations).

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem. Clarendon Press, Oxford, 1965.

Mathematical Background:

We know that a matrix A of order n can be reduced to almost triangular form by means of (n-2) successive unitary similarities (see description of subroutine MATU). Furthermore, when A is symmetric, these transformations preserve the property of symmetry, and the resulting matrix is symmetric and tridiagonal. Let us consider the sequence of such similarities that reduces A to the tridiagonal form $A^{(n-1)}$.

$$A^{(i+1)} = P_i A^{(i)} P_i, \quad A^{(1)} = A,$$

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

where $A^{(i)}$ is assumed to be of tridiagonal form in its first (i-1) rows and symmetric, and where P_i is the Householder matrix such that $A^{(i+1)}$ is of tridiagonal form in its first i rows. We know that P_i is defined by

$$P_i = I + \frac{1}{b(v_{i+1}-b)} (v - be_{i+1}) (v - be_{i+1})^T$$

where:

$$v^T = (v_1, v_2, \ldots, v_n)$$

$$v_k = 0, \text{ for } k = 1, 2, \ldots, i$$

$$v_k = a^{(i)}_{k,i}, \text{ for } k = i+1, \ldots, n$$

$$b = \pm < v, v >^{1/2}, \text{ sign (b) = - sign } (v_{i+1})$$

and where e_{i+1} is a vector whose (i+1)st component is one, the others being zero (see mathematical description of subroutine MATU).

Putting $x = v - be_{i+1}$ and $\alpha = \left[b (v_{i+1} - b)\right]^{-1}$, we have

$$P_i A^{(i)} P_i = A^{(i)} + \alpha A^{(i)} xx^T + \alpha xx^T A^{(i)}$$
$$+ \alpha^2 < x, A^{(i)} x > xx^T$$
$$= A^{(i)} + \left[A^{(i)} x + 1/2 < x, A^{(i)} x> \alpha x\right] \alpha x^T$$
$$+ \alpha x \left[x^T A^{(i)} + 1/2 < x, A^{(i)} x > \alpha x^T\right]$$

Since $A^{(i)} = A^{(i)T}$, this can be written as

$$P_i A^{(i)} P_i = A^{(i)} + YZ^T + ZY^T \qquad (1)$$

where

$$Y = \left[A^{(i)} + \frac{\alpha}{2} < x, A^{(i)} x > I\right] x \qquad (2)$$

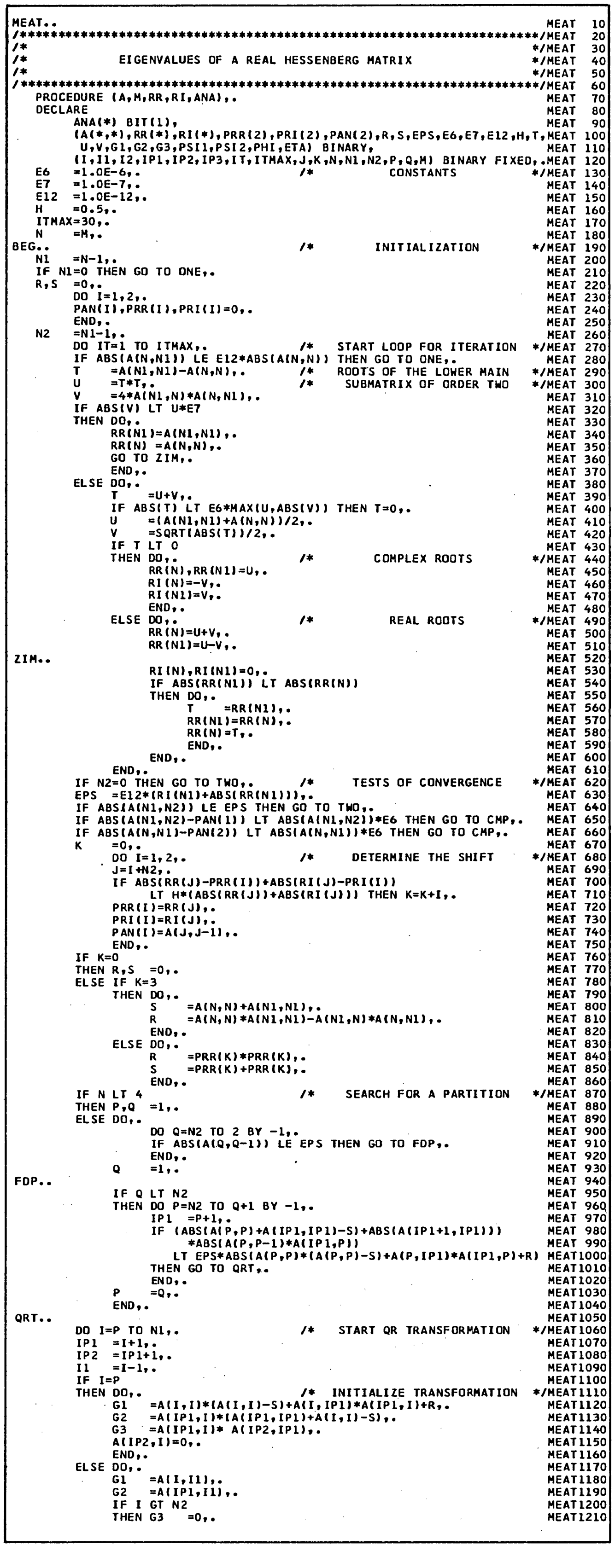
$$Z = \alpha x$$

Programming Considerations:

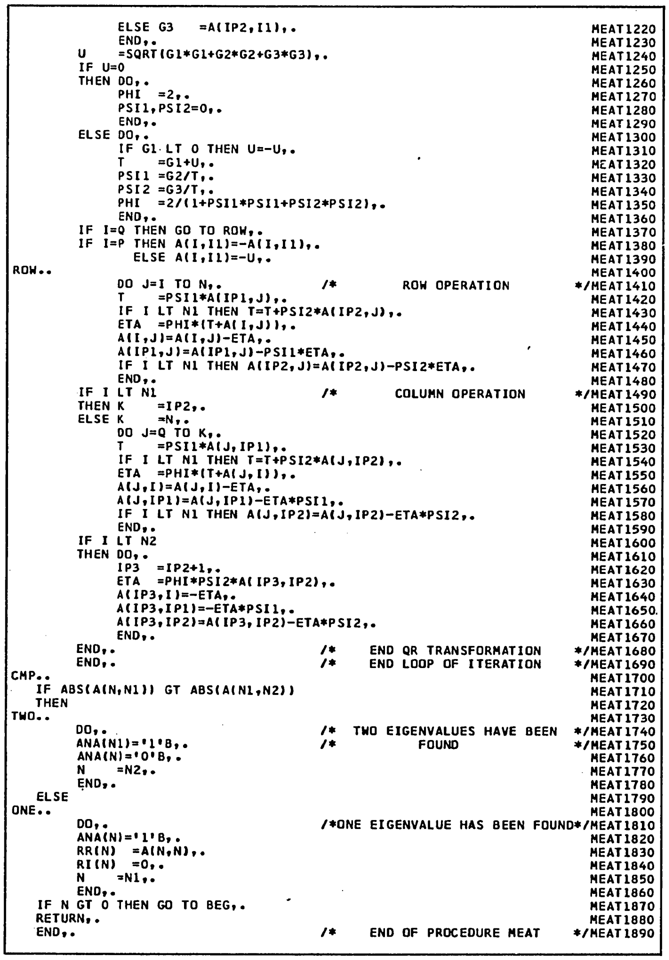
In the subroutine each similarity is performed on the upper part of the matrix according to equations (1) and (2).

The scalar products needed by the process are computed in double precision.

- Subroutine MEAT



```
MEAT..
/**********************************************************************/
/*                                                                    */
/*            EIGENVALUES OF A REAL HESSENBERG MATRIX                 */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,M,RR,RI,ANA),.
   DECLARE
        ANA(*) BIT(1),
        (A(*,*),RR(*),RI(*),PRR(2),PRI(2),PAN(2),R,S,EPS,E6,E7,E12,H,T,
         U,V,G1,G2,G3,PSI1,PSI2,PHI,ETA) BINARY,
        (I,I1,I2,IP1,IP2,IP3,IT,ITMAX,J,K,N,N1,N2,P,Q,M) BINARY FIXED,.
   E6   =1.0E-6,.                         /*          CONSTANTS          */
   E7   =1.0E-7,.
   E12  =1.0E-12,.
   H    =0.5,.
   ITMAX=30,.
   N    =M,.
BEG..                                     /*        INITIALIZATION       */
   N1   =N-1,.
   IF N1=0 THEN GO TO ONE,.
   R,S  =0,.
        DO I=1,2,.
        PAN(I),PRR(I),PRI(I)=0,.
        END,.
   N2   =N1-1,.
        DO IT=1 TO ITMAX,.                /*   START LOOP FOR ITERATION  */
        IF ABS(A(N,N1)) LE E12*ABS(A(N,N)) THEN GO TO ONE,.
        T    =A(N1,N1)-A(N,N),.           /*   ROOTS OF THE LOWER MAIN   */
        U    =T*T,.                       /*    SUBMATRIX OF ORDER TWO   */
        V    =4*A(N1,N)*A(N,N1),.
        IF ABS(V) LT U*E7
        THEN DO,.
             RR(N1)=A(N1,N1),.
             RR(N) =A(N,N),.
             GO TO ZIM,.
             END,.
        ELSE DO,.
             T    =U+V,.
             IF ABS(T) LT E6*MAX(U,ABS(V)) THEN T=0,.
             U    =(A(N1,N1)+A(N,N))/2,.
             V    =SQRT(ABS(T))/2,.
             IF T LT 0
             THEN DO,.                    /*        COMPLEX ROOTS        */
                  RR(N),RR(N1)=U,.
                  RI(N)=-V,.
                  RI(N1)=V,.
                  END,.
             ELSE DO,.                    /*          REAL ROOTS         */
                  RR(N)=U+V,.
                  RR(N1)=U-V,.
ZIM..
                  RI(N),RI(N1)=0,.
                  IF ABS(RR(N1)) LT ABS(RR(N))
                  THEN DO,.
                       T    =RR(N1),.
                       RR(N1)=RR(N),.
                       RR(N)=T,.
                       END,.
                  END,.
             END,.
        IF N2=0 THEN GO TO TWO,.          /*    TESTS OF CONVERGENCE     */
        EPS  =E12*(RI(N1)+ABS(RR(N1))),.
        IF ABS(A(N1,N2)) LE EPS THEN GO TO TWO,.
        IF ABS(A(N1,N2)-PAN(1)) LT ABS(A(N1,N2))*E6 THEN GO TO CMP,.
        IF ABS(A(N,N1)-PAN(2)) LT ABS(A(N,N1))*E6 THEN GO TO CMP,.
        K    =0,.
             DO I=1,2,.                   /*    DETERMINE THE SHIFT      */
             J=I+N2,.
             IF ABS(RR(J)-PRR(I))+ABS(RI(J)-PRI(I))
                  LT H*(ABS(RR(J))+ABS(RI(J))) THEN K=K+I,.
             PRR(I)=RR(J),.
             PRI(I)=RI(J),.
             PAN(I)=A(J,J-1),.
             END,.
        IF K=0
        THEN R,S  =0,.
        ELSE IF K=3
             THEN DO,.
                  S    =A(N,N)+A(N1,N1),.
                  R    =A(N,N)*A(N1,N1)-A(N1,N)*A(N,N1),.
                  END,.
             ELSE DO,.
                  R    =PRR(K)*PRR(K),.
                  S    =PRR(K)+PRR(K),.
                  END,.
        IF N LT 4                         /*   SEARCH FOR A PARTITION    */
        THEN P,Q  =1,.
        ELSE DO,.
                  DO Q=N2 TO 2 BY -1,.
                  IF ABS(A(Q,Q-1)) LE EPS THEN GO TO FDP,.
                  END,.
             Q    =1,.
FDP..
             IF Q LT N2
             THEN DO P=N2 TO Q+1 BY -1,.
                  IP1  =P+1,.
                  IF (ABS(A(P,P)+A(IP1,IP1)-S)+ABS(A(IP1+1,IP1)))
                        *ABS(A(P,P-1)*A(IP1,P))
                     LT EPS*ABS(A(P,P)*(A(P,P)-S)+A(P,IP1)*A(IP1,P)+R)
                  THEN GO TO QRT,.
                  END,.
             P    =Q,.
             END,.
QRT..
        DO I=P TO N1,.                    /*  START QR TRANSFORMATION    */
        IP1  =I+1,.
        IP2  =IP1+1,.
        I1   =I-1,.
        IF I=P
        THEN DO,.                         /* INITIALIZE TRANSFORMATION   */
             G1   =A(I,I)*(A(I,I)-S)+A(I,IP1)*A(IP1,I)+R,.
             G2   =A(IP1,I)*(A(IP1,IP1)+A(I,I)-S),.
             G3   =A(IP1,I)* A(IP2,IP1),.
             A(IP2,I)=0,.
             END,.
        ELSE DO,.
             G1   =A(I,I1),.
             G2   =A(IP1,I1),.
             IF I GT N2
             THEN G3   =0,.
```



```
             ELSE G3   =A(IP2,I1),.
             END,.
        U    =SQRT(G1*G1+G2*G2+G3*G3),.
        IF U=0
        THEN DO,.
             PHI  =2,.
             PSI1,PSI2=0,.
             END,.
        ELSE DO,.
             IF G1 LT 0 THEN U=-U,.
             T    =G1+U,.
             PSI1 =G2/T,.
             PSI2 =G3/T,.
             PHI  =2/(1+PSI1*PSI1+PSI2*PSI2),.
             END,.
        IF I=Q THEN GO TO ROW,.
        IF I=P THEN A(I,I1)=-A(I,I1),.
               ELSE A(I,I1)=-U,.
ROW..
             DO J=I TO N,.              /*        ROW OPERATION        */
             T    =PSI1*A(IP1,J),.
             IF I LT N1 THEN T=T+PSI2*A(IP2,J),.
             ETA  =PHI*(T+A(I,J)),.
             A(I,J)=A(I,J)-ETA,.
             A(IP1,J)=A(IP1,J)-PSI1*ETA,.
             IF I LT N1 THEN A(IP2,J)=A(IP2,J)-PSI2*ETA,.
             END,.
        IF I LT N1                      /*       COLUMN OPERATION      */
        THEN K    =IP2,.
        ELSE K    =N,.
             DO J=Q TO K,.
             T    =PSI1*A(J,IP1),.
             IF I LT N1 THEN T=T+PSI2*A(J,IP2),.
             ETA  =PHI*(T+A(J,I)),.
             A(J,I)=A(J,I)-ETA,.
             A(J,IP1)=A(J,IP1)-ETA*PSI1,.
             IF I LT N1 THEN A(J,IP2)=A(J,IP2)-ETA*PSI2,.
             END,.
        IF I LT N2
        THEN DO,.
             IP3  =IP2+1,.
             ETA  =PHI*PSI2*A(IP3,IP2),.
             A(IP3,I)=-ETA,.
             A(IP3,IP1)=-ETA*PSI1,.
             A(IP3,IP2)=A(IP3,IP2)-ETA*PSI2,.
             END,.
        END,.                           /*    END QR TRANSFORMATION    */
        END,.                           /*    END LOOP OF ITERATION    */
CMP..
   IF ABS(A(N,N1)) GT ABS(A(N1,N2))
   THEN
TWO..
        DO,.                            /*  TWO EIGENVALUES HAVE BEEN  */
        ANA(N1)='1'B,.                  /*         FOUND               */
        ANA(N)='0'B,.
        N    =N2,.
        END,.
   ELSE
ONE..
        DO,.                            /*ONE EIGENVALUE HAS BEEN FOUND*/
        ANA(N)='1'B,.
        RR(N)  =A(N,N),.
        RI(N)  =0,.
        N    =N1,.
        END,.
   IF N GT 0 THEN GO TO BEG,.
   RETURN,.
   END,.                                /*    END OF PROCEDURE MEAT    */
```

Purpose:

MEAT computes the eigenvalues of a real upper almost triangular matrix (Hessenberg form -- see subroutines MATE and MATU) using the double QR iteration.

Usage:

CALL MEAT (A, M, RR, RI, ANA);

A(M,M) - BINARY FLOAT
Given almost triangular matrix.

M - BINARY FIXED
Given order of the matrix.

RR(M) - BINARY FLOAT
Resultant vector containing the real parts of the eigenvalues.

RI(M) - BINARY FLOAT
Resultant vector containing the imaginary parts of the eigenvalues.

ANA(M) - BIT(1)
Resultant vector containing information for checking the results (see "Programming Considerations", below).

Remarks:

The original matrix is destroyed.

Method:

Double QR iteration of J. G. F. Francis

For reference see:

J. G. F. Francis, Computer Journal, October 1961, 4-3; January 1962, 4-4.
J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical Background:

1. Definition of the QR iteration

Let A be a real or complex nonsingular matrix of order n. Then a decomposition of A exists of the form

$$A = Q\,R$$

where Q is unitary and R is upper triangular. If the diagonal elements of R are real and positive, Q is unique. Consider now the sequence of matrices $A^{(p)}$ defined recursively by

$$A^{(o)} = A,\ A^{(p)} = Q^{(p)}\,R^{(p)},\ A^{(p+1)} = R^{(p)}Q^{(p)}{}_{p \geq 0}.$$

Note that $A^{(p+1)} = Q^{(p)*}\,A^{(p)}\,Q^{(p)}$ for $p \geq 0$; hence, $A^{(p)}$ is similar to A for all p.

Furthermore, if A satisfies certain conditions, it can be proved that $A^{(p)}$ tends to an upper triangular matrix as $p \to \infty$; thus the eigenvalues of A are the diagonal elements of this limit matrix.

2. Convergence

If the moduli of the eigenvalues are distinct, the elements $a_{ij}^{(p)}$ below the main diagonal of $A^{(p)}$ tend to zero, as do $|\lambda_i|^p / |\lambda_j|^p$, the eigenvalues being subscripted so that $|\lambda i| > |\lambda_i + 1|$.

Thus, in general, the eigenvalues appear on the main diagonal, starting from the last position, in increasing order of moduli.

So, when the smallest eigenvalue λ_n has been found, we can reduce the order of the matrix by neglecting the last row and column and find λ_{n-1} by the same process, without any special deflation.

Note that the speed of convergence is considerably improved when the origin of the eigenvalues is shifted close to λ_n.

Such a shift -- say, $s^{(p)}$ -- can be introduced before an iteration and the opposite one afterwards. Then the iteration can be written as:

$$A^{(p)} - s^{(p)}\,I = Q^{(p)}\,R^{(p)}$$

$$A^{(p+1)} = R^{(p)}\,Q^{(p)} + s^{(p)}\,I$$

In general, $A_{n,n}^{(p)}$, for p large enough, can provide an efficient value for $s^{(p)}$.

3. Use of the Hessenberg form

The Hessenberg form is preserved under the QR iteration. Thus, a reduction of the initial matrix to the Hessenberg form can give a significant saving of computation in each iteration for the QR decomposition, the lower part of the matrix consisting only of the codiagonal terms.

Before each iteration, the codiagonal terms will be inspected. If some of these are zero, the matrix will be split according to this occurrence, and the iteration will be applied to the lower main submatrix only.

4. The double QR iteration

Let A be a diagonalizable real upper Hessenberg matrix. Such a matrix must be expected to have complex conjugate pairs of eigenvalues. If these pairs are the only eigenvalues of equal modulus, it can be shown that they will appear as the latent roots of main submatrices of order 2. In this case, if a shift is close to one of these roots, it will be complex, and we will have to deal with complex matrices, although the initial one is real. The use of the double QR iteration avoids this inconvenience.

Taking $s^{(p+1)} = \bar{s}^{(p)}$, consider the transformation giving $A^{(p+2)}$ from $A^{(p)}$:

$$A^{(p+2)} = Q^{(p+1)*}\,Q^{(p)*}\,A^{(p)}\,Q^{(p)}\,Q^{(p+1)}$$

It can be proved that the product $Q^{(p)}Q^{(p+1)}$ derives from the QR decomposition of the matrix $M = (A^{(p)} - s^{(p)}\,I)\,(A^{(p)} - s^{(p+1)}\,I)$, which is real.

In fact, Francis (1961, 1962) showed that only the first column m_1 of M is necessary for determining the transformation which gives $A^{(p+2)}$ from $A^{(p)}$, if they both have the Hessenberg form.

Practically, the first part of the double iteration consists of the application of an initial transformation $N_1^*\,A^{(p)}\,N_1$ where N_1 is unitary and such that $N_1^*\,m_1 = \pm \|m_1\|\,e_1$. This leads to a matrix that no longer has the Hessenberg form.

Thus, the remaining part of the iteration will involve the application of (n-1) successive transformations, which have the same form as the initial one whose matrices N_i are such that the resulting matrix $A^{(p+2)}$ has the Hessenberg form.

This process can fail when a subdiagonal term of the given matrix is zero. In this case, the matrix can be split, and the iteration is performed on the lower main submatrix only.

In the subroutine, N_i are Householder's matrices.

Programming Considerations:

At each iteration, the latent roots x_1 and x_2 of the lower main submatrix of order 2 are computed. Then the following situations can occur:

1. The term $a_{n-1,n-2}$ can be taken as zero. Then x_1 and x_2 are eigenvalues of the original matrix, and the order of the matrix is reduced by 2. ANA(N) and ANA(N-1) are set to 0 and 1 respectively.

2. The term $a_{n,n-1}$ can be taken as zero. In this case, $a_{n,n}$ is an eigenvalue of the original matrix, and the order of the matrix is reduced by 1. ANA(N) is set to 1.

3. One of the last two subdiagonal terms is stable through one iteration. Then the smaller one is considered as zero. The corresponding components of ANA are set to 0 or 1, according to situation 1 or 2.

4. The maximum number of iterations is reached. In this case the smaller of the last two subdiagonal elements is taken as zero. The corresponding components of ANA are set to 0 or 1, according to situation 1 or 2.

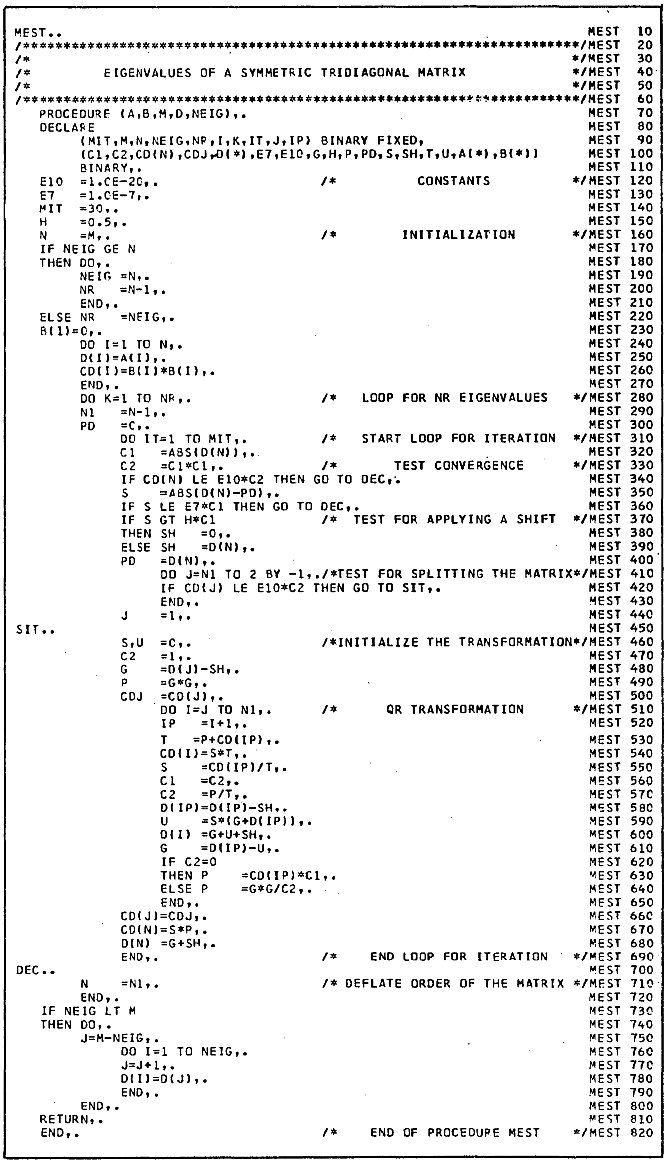
The user can check the results by inspecting the subdiagonal terms of the matrix on return from the subroutine, according to the vector ANA, in the following way: If, for each ANA(I) containing 1,

$$|A(I,I-1)| \leq 10^{-7} (|RR(I)| + |RI(I)|),$$

$$i = 2, \ldots, M$$

then RR(I) and RI(I) were computed with a satisfactory accuracy.

- Subroutine MEST



```
MEST..
/**********************************************************************/
/*                                                                    */
/*         EIGENVALUES OF A SYMMETRIC TRIDIAGONAL MATRIX              */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,B,M,D,NEIG),.
   DECLARE
        (MIT,M,N,NEIG,NR,I,K,IT,J,IP) BINARY FIXED,
        (C1,C2,CD(N),CDJ,D(*),E7,E10,G,H,P,PD,S,SH,T,U,A(*),B(*))
        BINARY,.
   E10  =1.CE-2C,.                    /*           CONSTANTS          */
   E7   =1.CE-7,.
   MIT  =30,.
   H    =0.5,.
   N    =M,.                          /*         INITIALIZATION       */
   IF NEIG GE N
   THEN DO,.
        NEIG =N,.
        NR   =N-1,.
        END,.
   ELSE NR   =NEIG,.
   B(1)=C,.
        DO I=1 TO N,.
        D(I)=A(I),.
        CD(I)=B(I)*B(I),.
        END,.
        DO K=1 TO NR,.                /*    LOOP FOR NR EIGENVALUES   */
        N1   =N-1,.
        PD   =C,.
             DO IT=1 TO MIT,.         /*    START LOOP FOR ITERATION  */
             C1   =ABS(D(N)),.
             C2   =C1*C1,.            /*        TEST CONVERGENCE      */
             IF CD(N) LE E10*C2 THEN GO TO DEC,.
             S    =ABS(D(N)-PD),.
             IF S LE E7*C1 THEN GO TO DEC,.
             IF S GT H*C1             /*   TEST FOR APPLYING A SHIFT  */
             THEN SH   =0,.
             ELSE SH   =D(N),.
             PD   =D(N),.
                  DO J=N1 TO 2 BY -1,./*TEST FOR SPLITTING THE MATRIX*/
                  IF CD(J) LE E10*C2 THEN GO TO SIT,.
                  END,.
             J    =1,.
SIT..
             S,U  =C,.                /*INITIALIZE THE TRANSFORMATION*/
             C2   =1,.
             G    =D(J)-SH,.
             P    =G*G,.
             CDJ  =CD(J),.
                  DO I=J TO N1,.      /*       QR TRANSFORMATION      */
                  IP   =I+1,.
                  T   =P+CD(IP),.
                  CD(I)=S*T,.
                  S    =CD(IP)/T,.
                  C1   =C2,.
                  C2   =P/T,.
                  D(IP)=D(IP)-SH,.
                  U    =S*(G+D(IP)),.
                  D(I) =G+U+SH,.
                  G    =D(IP)-U,.
                  IF C2=0
                  THEN P     =CD(IP)*C1,.
                  ELSE P     =G*G/C2,.
                  END,.
             CD(J)=CDJ,.
             CD(N)=S*P,.
             D(N) =G+SH,.
             END,.                    /*     END LOOP FOR ITERATION   */
DEC..
        N    =N1,.                    /* DEFLATE ORDER OF THE MATRIX */
        END,.
   IF NEIG LT M
   THEN DO,.
        J=M-NEIG,.
             DO I=1 TO NEIG,.
             J=J+1,.
             D(I)=D(J),.
             END,.
        END,.
   RETURN,.
   END,.                              /*     END OF PROCEDURE MEST    */
```

Purpose:

MEST computes the eigenvalues of a real symmetric tridiagonal matrix (see subroutine MSTU).

Usage:

CALL MEST (A, B, M, D, NEIG);

A(M) - BINARY FLOAT
Given vector containing the diagonal terms of the matrix.

B(M) - BINARY FLOAT
Given vector containing in positions 2, 3, . . . , M, the codiagonal terms of the matrix.

M - BINARY FIXED
Given order of the matrix.
D(M) - BINARY FLOAT
Resultant vector containing the eigenvalues.
NEIG - BINARY FIXED
Given number of eigenvalues required (see "Remarks").

Remarks:

When the eigenvalues are well separated, this procedure generally gives the NEIG eigenvalues of smallest moduli in the first NEIG positions of vector D.

Vectors A and B are preserved.

Method:

QR iteration modified by Kaiser and Ortega.

For reference see:

J. M. Ortega and H. F. Kaiser, "The LL^T and QR methods for symmetric tridiagonal matrices", Computer Journal, Volume 6, 1963, pp. 99-101.
J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical Background:

The general properties of the QR algorithm are given in the description of subroutine MEAT ("Mathematical Background", items 1 and 2). We recall them briefly here.

For a given diagonalizable matrix A of order n, the QR iteration is defined by:

$$A^{(0)} = A, \; A^{(p)} = Q^{(p)} R^{(p)}, \; A^{(p+1)} = R^{(p)} Q^{(p)}$$

where $Q^{(p)} R^{(p)}$ is a unitary-triangular factorization of $A^{(p)}$. A condition on $R^{(p)}$ is assumed to ensure the uniqueness of the factorization. If the eigenvalues have distinct moduli, for example, $|\lambda_i| > |\lambda_{i+1}|$ for i = 1, ..., n-1, then we have the following properties:

1. When p tends to infinity, $A^{(p)}$ tends to a triangular matrix and the eigenvalues of A appear on the main diagonal of $A^{(p)}$, starting from the last position in increasing order of moduli.

2. The symmetry and the tridiagonal structure of a matrix are preserved under the QR iteration.

3. If the origin of the eigenvalues is shifted close to λ_n before an iteration and shifted back afterwards, then the rate of convergence of $a^{(p)}_{n,n}$ to λ_n -- that is, the rate of convergence of $a^{(p)}_{n,i}$ to zero for i = 1, ..., n-1, can be considerably improved.

From the second property we can see that a preliminary reduction of a symmetric matrix to a similar tridiagonal form will give a significant saving of computation for each QR iteration.

From the first property we note that no special deflation is needed when λ_n has been found to sufficient accuracy; the last row and column of the matrix are neglected and the iteration is applied to the reduced matrix to obtain λ_{n-1}.

Let us consider a step of the iteration, denoted by

$$A = QR \;, \; A' = RQ$$

where A' is the iterate of A, the iteration superscript being dropped for clarity of notation. A and A' are symmetric tridiagonal matrices of order n. A will be fully defined by its diagonal terms a_i , i = 1, ..., n and its subdiagonal terms b_i , i = 2, . . . , n. The terms of A' will be denoted by a_i' , i = 1 . . . , n and b_i' , i = 2, . . . , n.

The reduction of A to R can be completed by premultiplication by (n-1) orthogonal matrices (Plane Rotations) Q_i , i = 1, . . . , n-1 of the form

$$Q_i = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_i & s_i & & \\ & & & -s_i & c_i & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{bmatrix} \qquad i = 1 , . . . , n - 1$$

c_i and s_i are the cosine and sine of an angle such that

$$R = Q_{n-1} \cdots Q_1 A$$

Then:

$$Q = Q_1^t \cdots Q_{n-1}^t$$

c_i and s_i are given by

$$c_i = \frac{p_i}{(p_i^2 + b_{i+1}^2)^{1/2}}$$

$$s_i = \frac{b_{i+1}}{(p_i^2 + b_{i+1})^{1/2}}$$

$$i = 1, \ldots, n-1 \qquad (1)$$

with

$$p_i = c_{i-1}\, a_i - s_{i-1}\, c_{i-2}\, b_{i+1}$$

and

$$c_{-1} = 0 \,,\; c_0 = 1 \,,\; s_0 = 0$$

R will be defined by:

$$r_{i,i} = c_i\, p_i + s_i\, b_{i+1} \,,\; i = 1 \,, \ldots, n-1$$

$$r_{n,n} = p_n$$

$$r_{1,2} = c_1 b_2 + s_1\, a_2 \qquad (2)$$

$$r_{i,\, i+1} = c_i\, c_{i-1}\, b_{i+1} + s_i\, a_{i+1} \,, \quad i = 2, \ldots, n-1$$

$$r_{i,\, i+2} = s_i\, b_{i+2},\; i = 1, \ldots, n-2$$

$$r_{i,j} = 0 \quad \text{for} \quad j > i+2$$

The post-multiplication of R by Q will provide A', according to:

$$a'_1 = c_1\, r_{1,1} + s_1\, r_{1,2}$$

$$a'_i = c_{i-1}\, c_i\, r_{i,i} + s_i\, r_{i,i+1} \quad i = 2, \ldots, n-1$$

$$a'_n - c_{n-1}\, r_{n,n} \qquad (3)$$

$$b'_{i+1} = s_i\, r_{i+1,i+1} \quad i = 1, \ldots, n-1$$

Formulas (2) and (3) can be combined in order to get A' directly from A. This avoids the computation of the square roots appearing in the expressions of c_i and s_i.

Then the final algorithm can be expressed as follows:

$$u_0 = 0,\; c_0 = 1,\; b_{n+1} = 0,\; a_{n+1} = 0$$

$$g_i = a_i - u_{i-1}$$

$$p_1^2 = g_i^2 \Big/ c_{i-1}^2 \quad \text{when} \quad c_{i-1} \neq 0$$

$$= c_{i-2}^2\, b_i^2 \quad \text{when} \quad c_{i-1} = 0$$

$$b'^2_i = s_{i-1}^2\, (p_i^2 + b_{i+1}^2) \quad \text{for } i > 1 \qquad (4)$$

$$s_i^2 = b_{i+1}^2 \Big/ (p_i^2 + b_{i+1}^2)$$

$$c_i^2 = p_i^2 \Big/ (p_i^2 + b_{i+1}^2)$$

$$u_i = s_i^2\, (g_i + a_{i+1})$$

$$a'_i = g_i + u_i$$

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

Programming Considerations:

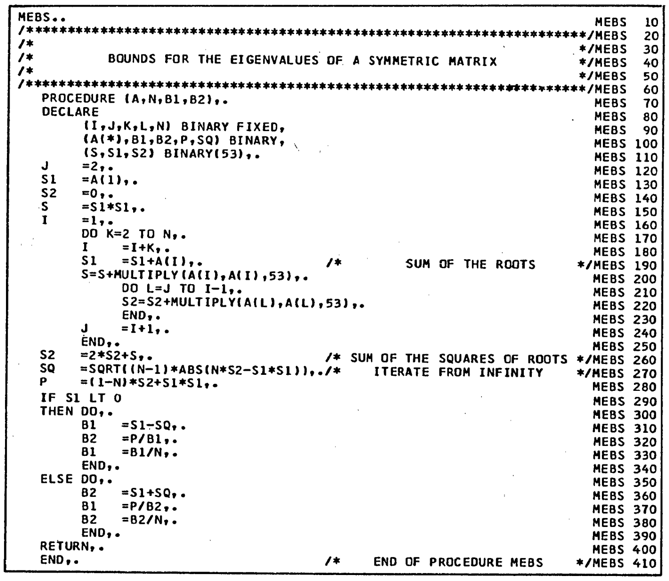
The iteration is performed according to equations (4). A shift of the origin of the eigenvalues is introduced in order to accelerate convergence. This shift is based on the last diagonal term of the matrix; it is applied only when convergence begins appearing.

When several eigenvalues are of same magnitude, codiagonal terms are close to zero. Then the matrix is split according to this occurrence and the iteration is performed on the lower main submatrix only. The iteration is stopped and the last diagonal term is taken as an eigenvalue when one of the following situations occurs:

1. The last subdiagonal term can be taken as zero.
2. The last subdiagonal term is stable through one iteration.
3. The maximum number of iterations is reached.

Then the order of the matrix is reduced by one and the process is repeated on the resulting matrix.

• Subroutine MEBS



```
MEBS..
/**********************************************************************/
/*                                                                    */
/*         BOUNDS FOR THE EIGENVALUES OF A SYMMETRIC MATRIX           */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,N,B1,B2),.
   DECLARE
        (I,J,K,L,N) BINARY FIXED,
        (A(*),B1,B2,P,SQ) BINARY,
        (S,S1,S2) BINARY(53),.
   J    =2,.
   S1   =A(1),.
   S2   =0,.
   S    =S1*S1,.
   I    =1,.
        DO K=2 TO N,.
        I    =I+K,.
        S1   =S1+A(I),.              /*      SUM OF THE ROOTS       */
        S=S+MULTIPLY(A(I),A(I),53),.
             DO L=J TO I-1,.
             S2=S2+MULTIPLY(A(L),A(L),53),.
             END,.
        J    =I+1,.
        END,.
   S2   =2*S2+S,.                    /* SUM OF THE SQUARES OF ROOTS */
   SQ   =SQRT((N-1)*ABS(N*S2-S1*S1)),./*   ITERATE FROM INFINITY    */
   P    =(1-N)*S2+S1*S1,.
   IF S1 LT 0
   THEN DO,.
        B1   =S1-SQ,.
        B2   =P/B1,.
        B1   =B1/N,.
        END,.
   ELSE DO,.
        B2   =S1+SQ,.
        B1   =P/B2,.
        B2   =B2/N,.
        END,.
   RETURN,.
   END,.                             /*   END OF PROCEDURE MEBS     */
```

Purpose:

MEBS computes a lower and an upper bound for the eigenvalues of a real symmetric matrix.

Usage:

CALL MEBS (A, N, B1, B2);

A (N*(N+1) /2) -	BINARY FLOAT Given real symmetric matrix in compressed storage mode.
N -	BINARY FIXED Given order of the matrix.
B1 -	BINARY FLOAT Resultant lower bound.
B2 -	BINARY FLOAT Resultant upper bound.

Method:

Laguerre's iteration is applied to the points at infinity.

For reference see:

B. Parlett, "Laguerre's Method Applied to the Matrix Eigenvalue Problem", Mathematics of Computation, 18, 1964.

Mathematical Background:

1. Laguerre's iteration.
   Let P(x) be a polynomial of degree n. The Laguerre iterate of a point x for the polynomial P can be expressed by

$$L_P(x) = x - \frac{n\,P(x)}{P'(x) \pm \sqrt{(n-1)\left[(n-1)\,P'(x)^2 - n\,P(x)\,P''(x)\right]}} \tag{1}$$

   Letting

$$S_1(x) = \frac{P'(x)}{P(x)} = \sum_{i=1}^{n} \frac{1}{x-x_i}$$

$$S_2(x) = \frac{P'(x)^2 - P(x)\,P''(x)}{P(x)^2}$$

$$= \sum_{i=1}^{n} \frac{1}{(x-x_i)^2}$$

where $x_1, \ldots, x_n$ are the roots of P(x), formula (1) can be written as

$$L_P(x) = x - \frac{n}{S_1 \pm \sqrt{(n-1)\,(nS_2 - S_1^2)}} \tag{2}$$

The sign of the square root is chosen so that the magnitude of the denominator is maximum.
When P(x) has real roots, we have the following properties:

   a. Let us consider a partition of the real line defined by the points at infinity and the zeros of P′(x). Starting from an initial point in any interval of the partition, the successive Laguerre iterates converge monotonically to the root therein. If the root is simple, convergence is asymptotically cubic.

   b. Laguerre's iterations are invariant under Möbius transformations.

2. Iterates of the points at infinity.
   From the first property of monotonic convergence, we can see that the iterates of the points at infinity will provide bounds for the roots. The second property gives the relation.

$$L_P(x) = \frac{1}{L_Q(\frac{1}{x})} \tag{3}$$

where Q is the polynomial reciprocal of P, the roots of which are

$$\frac{1}{x_i}, \; i = 1, \ldots, n.$$

Thus

$$L_P(\infty) = \frac{1}{L_Q(0)} \tag{4}$$

Now, if we combine equations (2) and (4), we can obtain the final formula

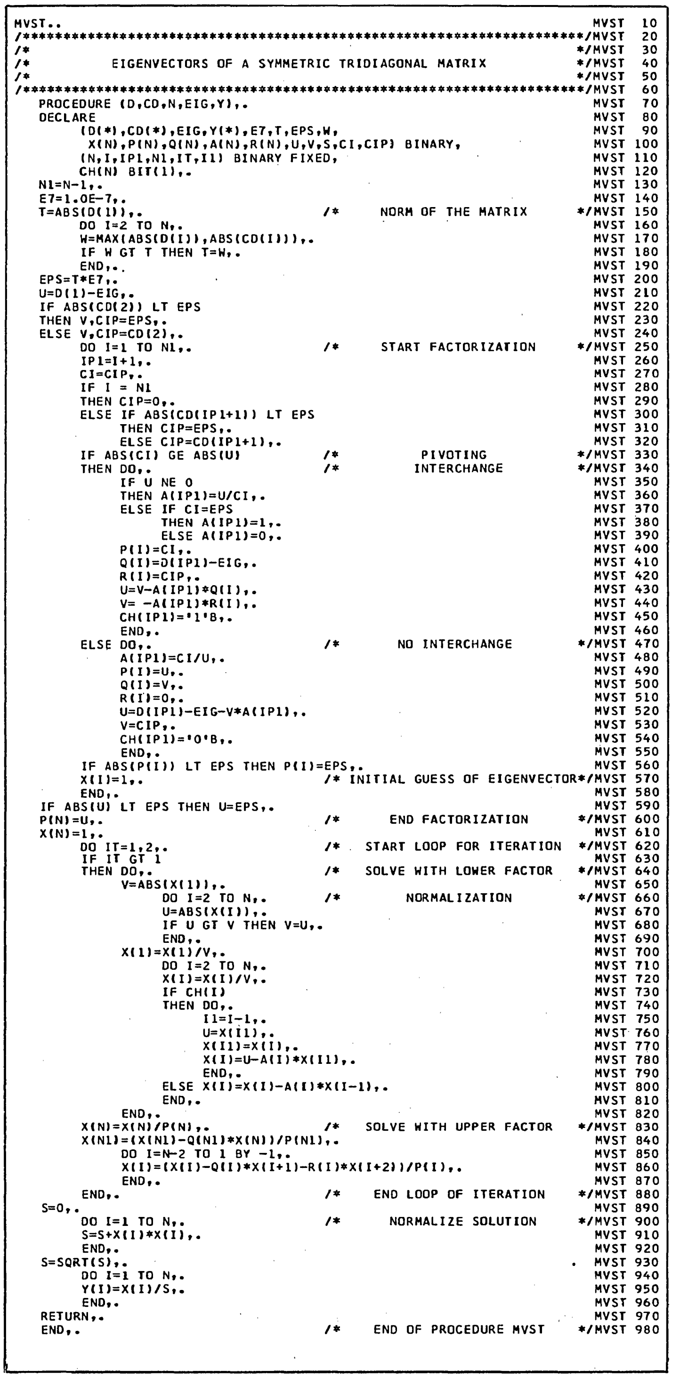
$$L_P(\infty) = \frac{1}{n}\left[\sigma_1 \pm \sqrt{(n-1)\;(n\,\sigma_2 - \sigma_1^2)}\right] \tag{5}$$

where σ_1 is the sum of the roots and σ_2 the sum of the squares of the roots of polynomial P.

Programming Considerations:

We can note that equation (5) does not require the coefficients of polynomial P but only the values of σ_1 and σ_2. If we apply this formula to the characteristic polynomial of a symmetric matrix (real roots), σ_1 will be obtained by computing the trace of the matrix and σ_2 the sum of the squares of the terms of the matrix. Then, equation (5) will give the bounds of the eigenvalues.

- Subroutine MVST



```
MVST..
/*********************************************************************/
/*                                                                   */
/*          EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX           */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (D,CD,N,EIG,Y),.
   DECLARE
        (D(*),CD(*),EIG,Y(*),E7,T,EPS,W,
         X(N),P(N),Q(N),A(N),R(N),U,V,S,CI,CIP) BINARY,
        (N,I,IP1,N1,IT,I1) BINARY FIXED,
        CH(N) BIT(1),.
   N1=N-1,.
   E7=1.0E-7,.
   T=ABS(D(1)),.                      /*     NORM OF THE MATRIX      */
        DO I=2 TO N,.
        W=MAX(ABS(D(I)),ABS(CD(I))),.
        IF W GT T THEN T=W,.
        END,..
   EPS=T*E7,.
   U=D(1)-EIG,.
   IF ABS(CD(2)) LT EPS
   THEN V,CIP=EPS,.
   ELSE V,CIP=CD(2),.
        DO I=1 TO N1,.                /*     START FACTORIZATION     */
        IP1=I+1,.
        CI=CIP,.
        IF I = N1
        THEN CIP=0,.
        ELSE IF ABS(CD(IP1+1)) LT EPS
             THEN CIP=EPS,.
             ELSE CIP=CD(IP1+1),.
        IF ABS(CI) GE ABS(U)          /*          PIVOTING           */
        THEN DO,.                     /*         INTERCHANGE         */
             IF U NE 0
             THEN A(IP1)=U/CI,.
             ELSE IF CI=EPS
                  THEN A(IP1)=1,.
                  ELSE A(IP1)=0,.
             P(I)=CI,.
             Q(I)=D(IP1)-EIG,.
             R(I)=CIP,.
             U=V-A(IP1)*Q(I),.
             V= -A(IP1)*R(I),.
             CH(IP1)='1'B,.
             END,.
        ELSE DO,.                     /*       NO INTERCHANGE        */
             A(IP1)=CI/U,.
             P(I)=U,.
             Q(I)=V,.
             R(I)=0,.
             U=D(IP1)-EIG-V*A(IP1),.
             V=CIP,.
             CH(IP1)='0'B,.
             END,.
        IF ABS(P(I)) LT EPS THEN P(I)=EPS,.
        X(I)=1,.                      /* INITIAL GUESS OF EIGENVECTOR*/
        END,.
   IF ABS(U) LT EPS THEN U=EPS,.
   P(N)=U,.                           /*      END FACTORIZATION      */
   X(N)=1,.
        DO IT=1,2,.                   /*   START LOOP FOR ITERATION  */
        IF IT GT 1
        THEN DO,.                     /*   SOLVE WITH LOWER FACTOR   */
             V=ABS(X(1)),.
                  DO I=2 TO N,.       /*        NORMALIZATION        */
                  U=ABS(X(I)),.
                  IF U GT V THEN V=U,.
                  END,.
             X(1)=X(1)/V,.
                  DO I=2 TO N,.
                  X(I)=X(I)/V,.
                  IF CH(I)
                  THEN DO,.
                       I1=I-1,.
                       U=X(I1),.
                       X(I1)=X(I),.
                       X(I)=U-A(I)*X(I1),.
                       END,.
                  ELSE X(I)=X(I)-A(I)*X(I-1),.
                  END,.
             END,.
        X(N)=X(N)/P(N),.              /*   SOLVE WITH UPPER FACTOR   */
        X(N1)=(X(N1)-Q(N1)*X(N))/P(N1),.
             DO I=N-2 TO 1 BY -1,.
             X(I)=(X(I)-Q(I)*X(I+1)-R(I)*X(I+2))/P(I),.
             END,.
        END,.                         /*    END LOOP OF ITERATION    */
   S=0,.
        DO I=1 TO N,.                 /*      NORMALIZE SOLUTION     */
        S=S+X(I)*X(I),.
        END,.
   S=SQRT(S),.
        DO I=1 TO N,.
        Y(I)=X(I)/S,.
        END,.
   RETURN,.
   END,.                              /*    END OF PROCEDURE MVST    */
```

Purpose:

For a given symmetric tridiagonal matrix, MVST provides the eigenvector corresponding to a given eigenvalue.

Usage:

CALL MVST (D, CD, N, EIG, Y);

D(N) - BINARY FLOAT
Given vector containing the diagonal terms of the matrix.

CD(N) - BINARY FLOAT
Given vector containing in positions 2, 3, ..., N the codiagonal terms of the matrix.

N - BINARY FIXED
Given order of the matrix.

EIG - BINARY FLOAT
Given eigenvalue.

Y(N) - BINARY FLOAT
Resultant vector containing the eigenvector.

Remarks:

Vectors D and CD remain unaltered.

Method:

Wielandt's inverse iteration is applied to the matrix, using the given eigenvalue as a shift.

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

J. H. Wilkinson, "Calculation of the eigenvectors of the symmetric tridiagonal matrix by inverse iteration", Numerische Mathematik, 4 (1962), pp. 368-376.

Mathematical Background:

Let us suppose that we know an approximation λ of an eigenvalue of a symmetric tridiagonal matrix A. A corresponding eigenvector V can be obtained by using Wielandt's inverse iteration (see the description of procedure MVAT), defined by the iterative process:

$$V^{(p+1)} = (A - \lambda I)^{-1} V^{(p)}$$

where $V^{(0)}$ is an arbitrary vector, not deficient in the eigenvector V.

Considering a triangular factorization of A-λI,

$$A - \lambda I = LR,$$

$V^{(p+1)}$ will be provided by solving successively the following equations:

$$LW = V^{(p)} \qquad (1)$$

$$RV^{(p+1)} = W \qquad (2)$$

When λ is close to an eigenvalue of A, $V^{(p)}$ tends very rapidly to V. Most of the time, two iterations are quite sufficient to provide an accurate approximation of V.

Programming Considerations:

A technique of partial pivoting by row interchange is used for the triangular factorization. This factorization is performed before starting the iterative process.

The two iterations are then carried out according to formulas (1) and (2).

The initial vector $V^{(0)}$ is chosen so that $V^{(0)} = Le$, with $e^T = (1, 1, \ldots, 1)$. Then the first iteration will consist of solving equation (2) only:

$$RV^{(1)} = e$$

- Subroutine MSDU

```
MSDU..
/*******************************************************************/
/*                                                                 */
/*      TO COMPUTE EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC */
/*      MATRIX                                                     */
/*                                                                 */
/*******************************************************************/
   PROCEDURE (A,R,N,MV),.
   DECLARE
        (I,IND,J,L,M,MV,N)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (A(*,*),R(*,*),ANORM,ANRMX,THR,U,Y,SINX,SINX2,COSX,COSX2,SINCS,
        FN)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                 */
   ERROR='0',.
   IF N LE 1                           /* THE ORDER OF MATRIX A IS    */
   THEN DO,.                           /* LESS THAN OR EQUAL TO ONE.  */
        ERROR='1',.
        GO TO FIN,.
        END,.
   FN   =N,.
   IF MV= 0
   THEN DO,.
             DO I = 1 TO N,.           /* GENERATE IDENTITY MATRIX    */
                  DO J = 1 TO N,.
                 R(I,J)=0,.
                  END,.
             R(I,I)=1,.
             END,.
        END,.
/*                                                                 */
/*      COMPUTE INITIAL AND FINAL NORM                             */
/*                                                                 */
   ANORM=0,.
        DO I = 1 TO N-1,.
             DO J = I+1 TO N,.
             ANORM=ANORM+A(I,J)*A(I,J),.
             END,.
        END,.
   IF ANORM LE C.0
   THEN GO TO SORT,.
   ANORM=1.414*SQRT(ANORM),.
   ANRMX=ANORM*1.0E-6/FN,.
/*                                                                 */
/*      INITIALIZE INDICATOR AND COMPUTE THRESHOLD, THR            */
/*                                                                 */
   IND  =C,.
   THR  =ANORM,.
S10..
   THR  =THR/FN,.
S20..
   L    =1,.
S30..
   M    =L+1,.
S40..
   IF ABS(A(L,M)) GE THR               /* COMPUTE SIN AND COS         */
   THEN DO,.
        IND  =1,.
        U    =0.5*(A(L,L)-A(M,M)),.
        Y    =-A(L,M)/SQRT(A(L,M)*A(L,M)+U*U),.
        IF U LT 0.0
        THEN Y    =-Y,.
        SINX =Y/SQRT(2.0*(1.0+(SQRT(1.0-Y*Y)))),.
        SINX2=SINX*SINX,.
        COSX =SQRT(1.0-SINX2),.
        COSX2=COSX*COSX,.
        SINCS=SINX*COSX,.
             DO I = 1 TO N,.           /* ROTATE L AND M COLUMNS      */
             IF I LT L
             THEN DO,.
                  IF I LT M
                  THEN DO,.
                       U    =A(I,L)*COSX-A(I,M)*SINX,.
                       A(I,M)=A(I,L)*SINX+A(I,M)*COSX,.
                       A(I,L)=U,.
                       END,.
                  END,.
             ELSE IF I GT L
             THEN DO,.
                  IF I LT M
                  THEN DO,.
                       U    =A(L,I)*COSX-A(I,M)*SINX,.
                       A(I,M)=A(L,I)*SINX+A(I,M)*COSX,.
                       END,.
                  ELSE IF I GT M
                  THEN DO,.
                       U    =A(L,I)*COSX-A(M,I)*SINX,.
                       A(M,I)=A(L,I)*SINX+A(M,I)*COSX,.
                       END,.
                  IF I NE M
                  THEN A(L,I)=U,.
                  END,.
             IF MV= 0
             THEN DO,.
                  U    =R(I,L)*COSX-R(I,M)*SINX,.
                  R(I,M)=R(I,L)*SINX+R(I,M)*COSX,.
                  R(I,L)=U,.
                  END,.
             END,.
        U    =2.C*A(L,M)*SINCS,.
        Y    =A(L,L)*COSX2+A(M,M)*SINX2-U,.
        U    =A(L,L)*SINX2+A(M,M)*COSX2+U,.
        A(L,M)=(A(L,L)-A(M,M))*SINCS+A(L,M)*(COSX2-SINX2),.
        A(L,L)=Y,.
        A(M,M)=U,.
        END,.
   IF M NE N                           /* TEST FOR M = LAST COLUMN    */
   THEN DO,.
        M    =M+1,.
        GO TO S40,.
        END,.
/*                                                                 */
/*      TEST FOR L = SECOND FROM LAST COLUMN                       */
/*                                                                 */
   IF L NE N-1
   THEN DO,.
        L    =L+1,.
```

```
        GO TO S30,.
        END,.
   IF IND= 1
   THEN DO,.
        IND  =0,.
        GO TO S20,.
        END,.
/*                                                                 */
/*      COMPARE THRESHOLD WITH FINAL NORM                          */
/*                                                                 */
   IF THR GT ANFMX
   THEN GO TO S10,.
/*                                                                 */
/*      SORT EIGENVALUES AND EIGENVECTORS                          */
/*                                                                 */
SORT..
        DO I = 1 TO N,.
             DO J = I TO N,.
             IF A(I,I) LT A(J,J)
             THEN DO,.
                  U    =A(I,I),.
                  A(I,I)=A(J,J),.
                  A(J,J)=U,.
                  IF MV= 0
                  THEN DO,.
                            DO L = 1 TO N,.
                            U    =R(L,I),.
                            R(L,I)=R(L,J),.
                            R(L,J)=U,.
                            END,.
                       END,.
                  END,.
             END,.
        END,.
FIN..
   RETURN,.
   END,.                            /*END OF PROCEDURE MSDU        */
```

Purpose:

MSDU computes eigenvalues and eigenvectors of a real symmetric matrix.

Usage:

CALL MSDU (A, R, N, MV);

A(N, N) - BINARY FLOAT [(53)]
Given matrix (symmetric), destroyed in computation.
Resultant eigenvalues are developed in the diagonal of matrix A in descending order.

R(N, N) - BINARY FLOAT [(53)]
Resultant matrix of eigenvectors (stored columnwise, in the same sequence as eigenvalues).

N - BINARY FIXED
Given order of matrix A and R.

MV - BINARY FIXED
Given code containing the following:
0--compute eigenvalues and eigenvectors.
1--compute eigenvalues only.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR=1 - The order of the matrix is one or less.

Note: If the initial norm is equal to zero, the matrix is diagonal.

Method:

Diagonalization method originated by Jacobi and adapted by Von Neumann for larger 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.

Mathematical Background:

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:

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

ν_I = initial norm

A = input matrix (symmetric)

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

The final norm is computed:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

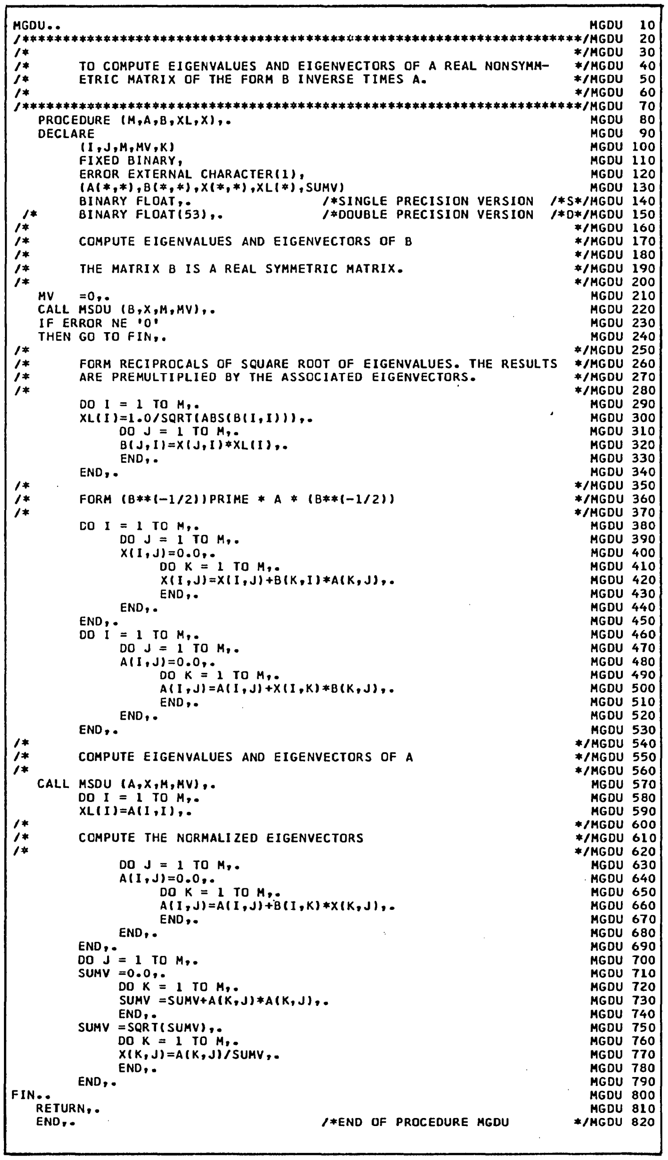
$$A_{lm} = (A_{ll} - A_{mm})\sin\theta\cos\theta + A_{lm}(\cos^2\theta - \sin^2\theta) \qquad (15)$$

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

Programming Considerations:

Matrix A cannot be in the same location as matrix R. If the eigenvectors are not calculated, the matrix R does not need to be dimensioned in the declare statement, but R must appear in the argument list of the procedure.

- Subroutine MGDU



```
MGDU..
/*********************************************************************/
/*                                                                    */
/*      TO COMPUTE EIGENVALUES AND EIGENVECTORS OF A REAL NONSYMM-    */
/*      ETRIC MATRIX OF THE FORM B INVERSE TIMES A.                   */
/*                                                                    */
/*********************************************************************/
   PROCEDURE (M,A,B,XL,X),.
   DECLARE
        (I,J,M,MV,K)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (A(*,*),B(*,*),X(*,*),XL(*),SUMV)
        BINARY FLOAT,.                /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
/*      COMPUTE EIGENVALUES AND EIGENVECTORS OF B                     */
/*                                                                    */
/*      THE MATRIX B IS A REAL SYMMETRIC MATRIX.                      */
/*                                                                    */
   MV   =0,.
   CALL MSDU (B,X,M,MV),.
   IF ERROR NE '0'
   THEN GO TO FIN,.
/*                                                                    */
/*      FORM RECIPROCALS OF SQUARE ROOT OF EIGENVALUES. THE RESULTS   */
/*      ARE PREMULTIPLIED BY THE ASSOCIATED EIGENVECTORS.             */
/*                                                                    */
        DO I = 1 TO M,.
        XL(I)=1.0/SQRT(ABS(B(I,I))),.
             DO J = 1 TO M,.
             B(J,I)=X(J,I)*XL(I),.
             END,.
        END,.
/*                                                                    */
/*      FORM (B**(-1/2))PRIME * A * (B**(-1/2))                       */
/*                                                                    */
        DO I = 1 TO M,.
             DO J = 1 TO M,.
             X(I,J)=0.0,.
                  DO K = 1 TO M,.
                  X(I,J)=X(I,J)+B(K,I)*A(K,J),.
                  END,.
             END,.
        END,.
        DO I = 1 TO M,.
             DO J = 1 TO M,.
             A(I,J)=0.0,.
                  DO K = 1 TO M,.
                  A(I,J)=A(I,J)+X(I,K)*B(K,J),.
                  END,.
             END,.
        END,.
/*                                                                    */
/*      COMPUTE EIGENVALUES AND EIGENVECTORS OF A                     */
/*                                                                    */
   CALL MSDU (A,X,M,MV),.
        DO I = 1 TO M,.
        XL(I)=A(I,I),.
/*                                                                    */
/*      COMPUTE THE NORMALIZED EIGENVECTORS                           */
/*                                                                    */
             DO J = 1 TO M,.
             A(I,J)=0.0,.
                  DO K = 1 TO M,.
                  A(I,J)=A(I,J)+B(I,K)*X(K,J),.
                  END,.
             END,.
        END,.
        DO J = 1 TO M,.
        SUMV =0.0,.
             DO K = 1 TO M,.
             SUMV =SUMV+A(K,J)*A(K,J),.
             END,.
        SUMV =SQRT(SUMV),.
             DO K = 1 TO M,.
             X(K,J)=A(K,J)/SUMV,.
             END,.
        END,.
FIN..
   RETURN,.
   END,.                           /*END OF PROCEDURE MGDU            */
```

Purpose:

MGDU computes eigenvalues and eigenvectors of a real matrix of the form B-inverse times A, where A is symmetric and B is positive definite.

Usage:

CALL MGDU (M, A, B, XL, X);

M - BINARY FIXED
Given order of square matrices A, B, and X.

A(M,M) - BINARY FLOAT [(53)]
Given symmetric matrix.

B(M,M) - BINARY FLOAT [(53)]
Given positive definite matrix.

XL(M) - BINARY FLOAT [(53)]
Resultant vector containing eigenvalues of B-inverse times A.

X(M,M) - BINARY FLOAT [(53)]
Resultant matrix containing eigenvectors columnwise.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero by the called subroutine MSDU. The following constitutes the possible error condition that may be detected:

ERROR=1 MSDU has been called and an error has occurred (see MSDU).

Subroutines and function subroutines required:

MSDU

Both matrices A and B are destroyed.

Method:

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

Mathematical Background:

This subroutine calculates the eigenvalues and the matrix of eigenvectors of the matrix B^{-1} A.

First the subroutine MSDU is used to calculate the eigenvalues and eigenvectors of the matrix B. The eigenvalues b_{ii} are stored in the main diagonal of the original matrix B and the eigenvectors are stored columnwise in the matrix X. Next the square roots of the reciprocals of the eigenvalues b_{ii} are formed and stored in XL

$$XL_i = 1/\sqrt{b_{ii}}$$

Then each eigenvector stored in X is multiplied by the corresponding value XL_j. The matrix of results is again stored in B. Next the matrix B^TAB is generated and stored in A. Then the subroutine MSDU is used to calculate the eigenvalues and eigenvectors of B^TAB. The eigenvalues are stored in XL and the eigenvectors are stored in X. Next the matrix product BX is formed and stored in A. The eigenvectors are then normalized to the form $a_{ij}/\sqrt{SUM_j a_{ij}^2}$ to form the desired output matrix of eigenvectors.

- Subroutine MVAT

```
MVAT..
/*********************************************************************/
/*                                                                   */
/*          EIGENVECTORS OF A COMPLEX HESSENBERG MATRIX              */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,N,EIG,V),.
   DECLARE
        P(N)
        BIT(1),
        (E7,U,T,EPS)
        BINARY,
        (A(*,*),EIG,C,V(*)).
        COMPLEX BINARY,
        S
        COMPLEX BINARY(53),
        (N,IS(N),I,I1,J,N1,K,K1,KP1,IT)
        BINARY FIXED,.
   E7=1.0E-7,.
   A(1,1)=A(1,1)-EIG,.                /*   MODIFY DIAGONAL ELEMENTS  */
   IS(1)=1,.
   U=ABS(A(1,1)),.                    /*COMPUTE A NORM OF THE MATRIX */
        DO I=2 TO N,.
        I1=I-1,.
        IS(I)=I1,.
        A(I,I)=A(I,I)-EIG,.
        T=ABS(A(1,I)),.
        IF T GT U THEN U=T,.
             DO J=I1 TO N,.
             T=ABS(A(I,J)),.
             IF T GT U THEN U=T,.
             END,.
        END,.
   EPS=U*E7,.
   N1=N-1,.                           /*     START FACTORIZATION     */
   P(1)='0'B,.
   IF ABS(A(2,1)) GT ABS(A(1,1))      /*       INITIALIZATION        */
   THEN DO,.
        P(1)='1'B,.
             DO I=1 TO N,.
             C=A(1,I),.
             A(1,I)=A(2,I),.
             A(2,I)=C,.
             END,.
        END,.
   IF ABS(A(1,1)) LT EPS THEN A(1,1)=EPS,.
   A(2,1)=A(2,1)/A(1,1),.
        DO K=2 TO N1,.
        KP1=K+1,.
        K1=K-1,.
        S=A(K,K),.                    /*   COMPUTE THE LOWER FACTOR  */
             DO I=IS(K) TO K1,.
             S=S-MULTIPLY(A(K,I),A(I,K),53),.
             END,.
        A(K,K)=S,.
        IF ABS(A(K,K)) LT ABS(A(KP1,K))
        THEN DO,.                     /*          PIVOTING           */
             P(K)='1'B,.
                  DO I=K TO N,.
                  C=A(K,I),.
                  A(K,I)=A(KP1,I),.
                  A(KP1,I)=C,.
                  END,.
                  DO I=IS(K) TO K1,.
                  A(KP1,I)=A(K,I),.
                  END,.
             I=IS(K),.
             IS(K)=IS(KP1),.
             IS(KP1)=I,.
             END,.
        ELSE DO,.
             P(K)='0'B,.
                  DO J=KP1 TO N,.     /*   COMPUTE THE UPPER FACTOR  */
                  S=A(K,J),.
                       DO I=IS(K) TO K1,.
                       S=S-MULTIPLY(A(I,J),A(K,I),53),.
                       END,.
                  A(K,J)=S,.
                  END,.
             END,.
                                      /*  NORMALIZE THE LOWER FACTOR */
        IF ABS(A(K,K)) LT EPS THEN A(K,K)=EPS,.
        A(KP1,K)=A(KP1,K)/A(K,K),.
        END,.
   S=A(N,N),.
        DO I=IS(N) TO N1,.
        S=S-MULTIPLY(A(N,I),A(I,N),53),.
        END,.
   A(N,N)=S,.                         /*      END FACTORIZATION      */
   IF ABS(A(N,N)) LT EPS THEN A(N,N)=EPS,.
        DO I=1 TO N,.                 /*      INVERSE ITERATION      */
        V(I)=1,.                      /*       STARTING VALUE        */
        END,.
        DO IT=1,2,.
        K=N,.
        IF IT GT 1
        THEN DO,.
                  DO I=1 TO N1,.      /*        INTERCHANGES         */
                  IF P(I)
                  THEN DO,.
                       I1=I+1,.
                       C=V(I),.
                       V(I)=V(I1),.
                       V(I1)=C,.
                       END,.
                  END,.
                  DO I=2 TO N,.       /*   SOLVE WITH LOWER FACTOR   */
                       S=V(I),.
                            DO J=IS(I) TO I-1,.
                            S=S-MULTIPLY(A(I,J),V(J),53),.
                            END,.
                       V(I)=S,.
                  END,.
             END,.
        V(N)=V(N)/A(N,N),.            /*   SOLVE WITH UPPER FACTOR   */
        U=ABS(V(N)),.
             DO I=N1 TO 1 BY -1,.
             S=V(I),.
                  DO J=I+1 TO N,.
                  S=S-MULTIPLY(A(I,J),V(J),53),.
                  END,.
             V(I)=S/A(I,I),.
             T=ABS(V(I)),.
             IF T GT U
             THEN DO,.
                  K=I,.
                  U=T,.
                  END,.
             END,.
        C    =V(K),.
             DO I=1 TO N,.            /*  NORMALIZE RESULTING VECTOR */
             V(I) =V(I)/C,.
             END,.
        END,.                         /*  END OF LOOP FOR ITERATION  */
   RETURN,.
   END,.                              /*    END OF PROCEDURE MVAT    */
```

Purpose:

For a given almost triangular complex matrix (Hessenberg), this procedure provides the eigenvector corresponding to a given eigenvalue.

Usage:

CALL MVAT (A, N, EIG, V);

A(N,N) - COMPLEX BINARY FLOAT
Given almost triangular matrix.
N - BINARY FIXED
Given order of the matrix.
EIG - COMPLEX BINARY FLOAT
Given eigenvalue.
V(N) - COMPLEX BINARY FLOAT
Resultant vector containing the eigenvector corresponding to EIG.

Remarks:

The original matrix is destroyed.

Method:

Wielandt's inverse iteration is applied to the matrix, using the given eigenvalue as a shift.

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical Background:

For a given nonsingular matrix A, the inverse iteration is defined by the following process:

$$V^{(p+1)} = A^{-1} V^{(p)}$$

where $V^{(0)}$ is an arbitrary starting vector. We know that when $P \rightarrow \infty$, under certain conditions $V^{(p)}$ tends to an eigenvector V associated with the smallest eigenvalue λ_0 of the matrix A.

When converging to V, the speed of convergence can be substantially improved by shifting the origin of the eigenvalues close to λ_0. Then the iteration can be written as

$$V^{(p + 1)} = (A - \lambda I)^{-1} V^{(p)} \tag{1}$$

where λ is the value of the shift.

When we know an approximation λ of λ_0, the above properties of the inverse iteration can be used for finding the corresponding eigenvector V by means of equation (1).

The closer λ is to λ_0, the faster $V^{(p)}$ converges to V. If λ has been obtained with good accuracy, V can be obtained using only a few steps of inverse iteration.

Each step of iteration is equivalent to finding the solution of the equation

$$(A - \lambda I) \; V^{(p+1)} = V^{(p)} \tag{2}$$

Considering a triangular factorization of $A - \lambda I$, $A - \lambda I = LR$, the solution of equation (2) will be provided by solving successively

$$LW = V^{(p)} \tag{3}$$

$$RV^{(p+1)} = W \tag{4}$$

where L and R are lower and upper triangular matrices. The triangular decomposition has to be performed only once before starting the iterative process, and the iteration is carried out by solving equations (3) and (4).

Programming Considerations:

A technique of partial pivoting by row interchange is included in the process of triangular factorization. This pivoting is obviously convenient in two ways; it is economical and does not modify the special structure of the matrix. Thus, it will be possible to take advantage of this structure in the factorization of the matrix, as well as in the solution of equation (3).

Since the starting vector is arbitrary, we choose it so that

$$V^{(0)} = Le, \; W = e,$$

where:

$$e^T = (1, 1, \ldots, 1)$$

Then the first iteration will consist of solving equation (4) only:

$$RV^{(1)} = e$$

Only two iterations are performed. Most of the time they are quite sufficient to provide an accurate approximation of the eigenvector V.

Subroutine MVSU

```
MVSU..
/*********************************************************************/
/*                                                                   */
/*              BACK TRANSFORMATION OF THE EIGENVECTORS              */
/*                           SYMMETRIC CASE                          */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,N,CD,V),.
   DECLARE
        (A(*),CD(*),V(*),T,C) BINARY,
        (M,N,ICD,K,KP1,KP2,J,I,L) BINARY FIXED,
        (S,DP) BINARY(53),.
   ICD=(N*(N+1))/2-1,.
        DO K=N-1 TO 2 BY -1,.
        KP1=K+1,.
        ICD=ICD-KP1,.
        C=A(ICD)-CD(K),.
        IF C NE 0
        THEN DO,.                         /*  ORTHOGONAL TRANSFORMATION  */
             S=0,.
                  J=ICD-K+1,.
                       DO I=K TO N,.
                       J=J+I-1,.
                       S=S+MULTIPLY(A(J),V(I),53),.
                       END,.
                  S=S/CD(K),.
                  T=(S-V(K))/C,.
                  V(K)=S,.
                  J=ICD,.
                       DO I=KP1 TO N,.
                       J=J+I-1,.
                       V(I)=V(I)+T*A(J),.
                       END,.
             END,.
        END,.
        S=0,.                             /*         NORMALIZE           */
             DO I=1 TO N,.
             DP=V(I),.
             S=S+DP*DP,.
             END,.
        S=SQRT(S),.
             DO I=1 TO N,.
             V(I)=V(I)/S,.
             END,.
    RETURN,.
    END,.                                 /*   END OF PROCEDURE MVSU   */
```

Purpose:

For a given symmetric matrix M that has been reduced to a similar tridiagonal symmetric matrix H by procedure MSTU, MVSU gives the eigenvector of M corresponding to a given eigenvector of H.

Usage:

CALL MVSU (A, N, CD, V);

A(N*(N+1)/2) - BINARY FLOAT
Given vector whose elements are set up by procedure MSTU.

N - BINARY FIXED
Given order of the original matrix M.

CD(N) - BINARY FLOAT
Given vector containing in positions 2, 3, ..., N the codiagonal terms of the tridiagonal matrix.

V(N) - BINARY FLOAT
Given eigenvector of the tridiagonal matrix. Resultant eigenvector of the original matrix.

Remarks:

See procedure MSTU.

Method:

The eigenvector of the almost triangular matrix H is transformed according to the unitary similarities applied to matrix M in procedure MSTU.

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical background:

For a symmetric matrix M of order n that has been reduced to the tridiagonal matrix H by similarities, we have a relation of the form

$$H = P^{-1} MP$$

and an eigenvector of M, X(M) corresponding to an eigenvector of H, X(H) according to

$$H(M) = P \cdot X(H) \qquad (1)$$

In procedure MSTU, P consists of the product of (n-2) Householder's matrices:

$$P = P_1 \cdot P_2 \cdots\cdots P_{n-2} \qquad (2)$$

$$P_i = I + \frac{1}{b(v_{i+1} - b)} (v - be_{i+1}) (v - be_{i+1})^T$$

where the vector v and the scalar b have been defined in the transformation of the i-th column of the given matrix in procedure MSTU.

P will thus be applied to X(H) by means of (n-2) successive transformations, P_{n-2}, P_{n-1}, ..., P_1, according to equations (1) and (2).

The elements v and b defining each P_i are transmitted to MVSU through the parameters A and B.

• Subroutine MVUB

```
MVUB..
/*******************************************************************/
/*                                                                 */
/*           BACK TRANSFORMATION OF THE EIGENVECTORS              .*/
/*                HOUSEHOLDER'S TRANSFORMATIONS                    */
/*                                                                 */
/*******************************************************************/
   PROCEDURE (A,N,B,V),.
   DECLARE
        (A(*,*),B(*),T,U) BINARY,
        (I,K,K1,KP1,N) BINARY FIXED,
        (V(*),X) COMPLEX BINARY,
        S COMPLEX BINARY(53),.
        DO K=N-1 TO 2 BY -1,.
        IF B(K) NE 0
        THEN DO,.                       /*  ORTHOGONAL TRANSFORMATION  */
             KP1=K+1,.
             K1=K-1,.
             S=MULTIPLY(B(K),V(K),53),.
                  DO I=KP1 TO N,.
                  S=S+MULTIPLY(A(I,K1),V(I),53),.
                  END,.
             S=S/A(K,K1),.
             X=(S-V(K))/(B(K)-A(K,K1)),.
             V(K)=S,.
                  DO I=KP1 TO N,.
                  V(I)=V(I)+X*A(I,K1),.
                  END,.
             END,.
        END,.
   K=1,.
   T=ABS(V(1)),.                        /*        NORMALIZE            */
        DO I=2 TO N,.
        U=ABS(V(I)),.
        IF U GT T
        THEN DO,.
             T=U,.
             K=I,.
             END,.
        END,.
   X    =V(K),.
        DO I=1 TO N,.
        V(I) =V(I)/X,.
        END,.
   RETURN,.
   END,.                                /*   END OF PROCEDURE MVUB     */
```

Purpose:

For a given matrix M that has been reduced to a similar almost triangular matrix H by procedure MATU, MVUB gives the eigenvector of M corresponding to a given eigenvector of H.

Usage:

CALL MVUB (A, N, B, V);

A(N, N) - BINARY FLOAT
Given two-dimensional array whose elements are set up by procedure MATU.
N - BINARY FIXED
Given order of the matrix.
B(N) - BINARY FLOAT
Given vector whose components are provided by procedure MATU.
V(N) - COMPLEX BINARY FLOAT
Given eigenvector of the almost triangular matrix.
Resultant eigenvector of the original matrix.

Remarks:

See procedure MATU.

Method:

The eigenvector of the tridiagonal matrix H is transformed according to the unitary similarities applied to the matrix M in procedure MATU.

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical background:

For a matrix M of order n that has been reduced to the almost triangular matrix H by similarities, we have a relation of the form

$$H = P^{-1}MP$$

and an eigenvector of M, X(M) corresponding to an eigenvector of H, X(H) according to

$$X(M) = P \cdot X(H) \qquad (1)$$

In procedure MATU, P consists of a product of (n-2) Householder's matrices:

$$P = P_1 \cdot P_2 \cdot\cdot\cdot\cdot\cdot P_{n-2} \qquad (2)$$

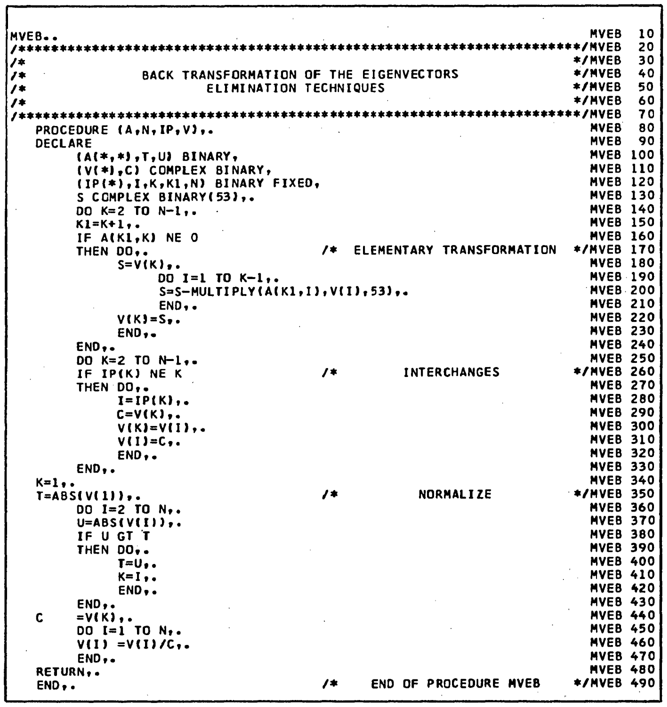
$$P_i = I + \frac{1}{v(v_{i+1} - b)} (v - be_{i+1}) (v - be_{i+1})^t$$

where the vector v and the scalar b have been defined in the transformation of the i-th column of the given matrix in procedure MATU.

P will thus be applied to X(H) by means of (n-2) successive transformations, $P_{n-2}, P_{n-1}, \dots, P_1$, according to equations (1) and (2).

The elements v and b defining each P_i are transmitted to MSTU through the parameters A and B.

• Subroutine MVEB



```
MVEB..
/**********************************************************************/
/*                                                                    */
/*             BACK TRANSFORMATION OF THE EIGENVECTORS                */
/*                    ELIMINATION TECHNIQUES                          */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,N,IP,V),.
   DECLARE
        (A(*,*),T,U) BINARY,
        (V(*),C) COMPLEX BINARY,
        (IP(*),I,K,K1,N) BINARY FIXED,
        S COMPLEX BINARY(53),.
        DO K=2 TO N-1,.
        K1=K+1,.
        IF A(K1,K) NE 0
        THEN DO,.                      /*  ELEMENTARY TRANSFORMATION  */
             S=V(K),.
                  DO I=1 TO K-1,.
                  S=S-MULTIPLY(A(K1,I),V(I),53),.
                  END,.
             V(K)=S,.
             END,.
        END,.
        DO K=2 TO N-1,.
        IF IP(K) NE K                  /*        INTERCHANGES         */
        THEN DO,.
             I=IP(K),.
             C=V(K),.
             V(K)=V(I),.
             V(I)=C,.
             END,.
        END,.
   K=1,.
   T=ABS(V(1)),.                       /*         NORMALIZE           */
        DO I=2 TO N,.
        U=ABS(V(I)),.
        IF U GT T
        THEN DO,.
             T=U,.
             K=I,.
             END,.
        END,.
   C    =V(K),.
        DO I=1 TO N,.
        V(I) =V(I)/C,.
        END,.
   RETURN,.
   END,.                               /*   END OF PROCEDURE MVEB     */
```

Purpose:

For a given matrix M that has been transformed to a similar almost triangular matrix H by procedure MATE, MVEB gives the eigenvector of M corresponding to a given eigenvector of H.

Usage:

CALL MVEB (A, N, IP, V);

A(N, N) - BINARY FLOAT
Given two-dimensional array whose elements are set up by procedure MATE.

N - BINARY FIXED
Given order of the almost triangular matrix.

IP(N) - BINARY FIXED
Given vector whose components are provided by procedure MATE.

V(N) - COMPLEX BINARY FLOAT
Given eigenvector of the almost triangular matrix.
Resultant eigenvector of the original matrix.

Remarks:

See procedures MATE and MVAT.

Method:

The eigenvector of the almost triangular matrix is transformed according to the similarities applied to the matrix M in procedure MATE.

For reference see:

J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Mathematical background:

We know that a given matrix M of order n can be reduced by similarity to an almost triangular matrix H. This can be written as

$$H = SMS^{-1}$$

Then, for a given eigenvalue of both M and H, the corresponding eigenvectors V of M and W of H are related by the equation

$$V = S^{-1}W$$

The transformation S is defined here as the product of a triangular matrix T with unit diagonal by a permutation matrix P which was operating on the rows of M according to the pivoting used in procedure MATE.

The elements of the matrix T are transmitted to the procedure through the array A. The permutation matrix P is defined by the information contained in vector IP.

Then V is provided by

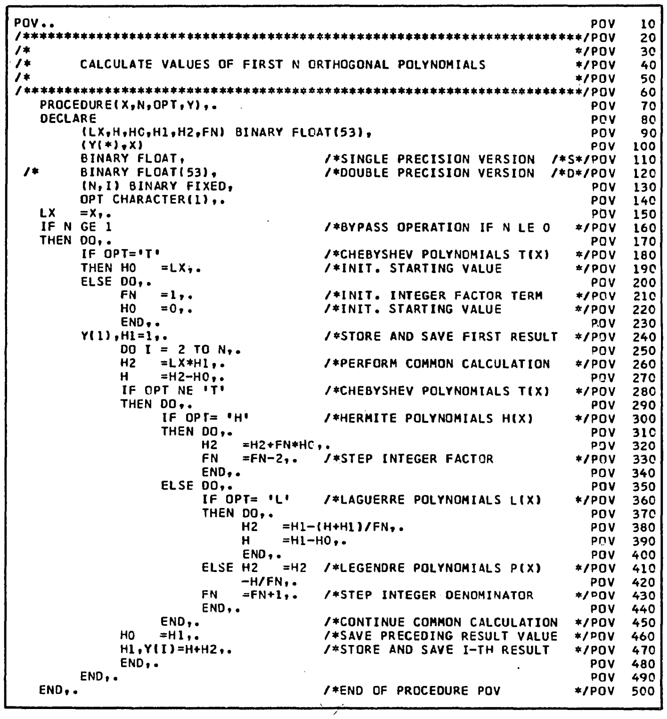
$$V = PX$$

where the vector X is the solution of the equation

$$TX = W$$

Polynomial Operations

- Subroutine POV



```
POV..
/**********************************************************************/
/*                                                                    */
/*      CALCULATE VALUES OF FIRST N ORTHOGONAL POLYNOMIALS            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,N,OPT,Y),.
   DECLARE
        (LX,H,HC,H1,H2,FN) BINARY FLOAT(53),
        (Y(*),X)
        BINARY FLOAT,                /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),            /*DOUBLE PRECISION VERSION  /*D*/
        (N,I) BINARY FIXED,
        OPT CHARACTER(1),.
   LX   =X,.
   IF N GE 1                         /*BYPASS OPERATION IF N LE 0   */
   THEN DO,.
        IF OPT='T'                   /*CHEBYSHEV POLYNOMIALS T(X)   */
        THEN HO   =LX,.              /*INIT. STARTING VALUE         */
        ELSE DO,.
             FN   =1,.               /*INIT. INTEGER FACTOR TERM    */
             HO   =0,.               /*INIT. STARTING VALUE         */
             END,.
        Y(1),H1=1,.                  /*STORE AND SAVE FIRST RESULT  */
             DO I = 2 TO N,.
             H2   =LX*H1,.           /*PERFORM COMMON CALCULATION   */
             H    =H2-HO,.
             IF OPT NE 'T'           /*CHEBYSHEV POLYNOMIALS T(X)   */
             THEN DO,.
                  IF OPT= 'H'        /*HERMITE POLYNOMIALS H(X)     */
                  THEN DO,.
                       H2   =H2+FN*HC,.
                       FN   =FN-2,.  /*STEP INTEGER FACTOR          */
                       END,.
                  ELSE DO,.
                       IF OPT= 'L'   /*LAGUERRE POLYNOMIALS L(X)    */
                       THEN DO,.
                            H2   =H1-(H+H1)/FN,.
                            H    =H1-HO,.
                            END,.
                       ELSE H2   =H2  /*LEGENDRE POLYNOMIALS P(X)   */
                            -H/FN,.
                       FN   =FN+1,.  /*STEP INTEGER DENOMINATOR     */
                       END,.
                  END,.              /*CONTINUE COMMON CALCULATION  */
             HO   =H1,.              /*SAVE PRECEDING RESULT VALUE  */
             H1,Y(I)=H+H2,.          /*STORE AND SAVE I-TH RESULT   */
             END,.
        END,.
   END,.                             /*END OF PROCEDURE POV         */
```

Purpose:

POV computes the values of the first n orthogonal polynomials. The user has the choice of

Chebyshev polynomials (T_0, T_1, ..., T_{n-1}) with OPT = 'T'
Legendre polynomials (P_0, P_1, ..., P_{n-1}) with OPT = 'P'
Laguerre polynomials (L_0, L_1, ..., L_{n-1}) with OPT = 'L'
Hermite polynomials (H_0, H_1, ..., H_{n-1}) with OPT = 'H'

Usage:

CALL POV (X, N, OPT, Y);

X - BINARY FLOAT [(53)]
Given argument of the orthogonal polynomials
N - BINARY FIXED
Given number of orthogonal polynomials to be calculated.
OPT - CHARACTER (1)
Given parameter of choice (see "Purpose").
Y(N) - BINARY FLOAT [(53)]
Resultant vector containing the values of the first N orthogonal polynomials.

Remarks:

Operation is bypassed if N is not positive. Any input value of OPT other than 'T', 'L', or 'H' is treated as if it were 'P'. The values of the shifted polynomials of Chebyshev or Legendre for argument x are obtained as values of non-shifted polynomials for the argument (2 • x - 1).

Method:

Evaluation is based on the three-term recurrence relation for orthogonal polynomials.

For reference see:

Jahnke-Emde-Loesch, Tables of Higher Functions, B. G. Teubner, Stuttgart, 1960, pp. 96-114.
M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, Applied Mathematics Series 55, National Bureau of Standards, 1964, pp. 771-803.

Mathematical Background:

The orthogonal polynomials are defined by the following iteration scheme:

Chebyshev polynomials $T_k(x)$

$T_0(x) = 1$

$T_1(x) = x$

$T_{k+1}(x) = 2x\,T_k(x) - T_{k-1}(x)$, for k = 1, 2, ...

Laguerre polynomials $P_k(x)$

$P_0(x) = 1$

$P_1(x) = x$

$(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x)$, for k = 1, 2, ...

Laguerre polynomials $L_k(x)$

$L_0(x) = 1$

$L_1(x) = 1 - x$

$(k+1)L_{k+1}(x) = (2k+1 - x)\ L_k\ (x) - kL_{k-1}(x)$,

for k = 1, 2, . . .

Hermite polynomials $\qquad H_k(x)$

$H_0(x) = 1$

$H_1(x) = 2x$

$H_{k+1}(x) = 2xH_k(x) - 2kH_{k-1}(x)$, for k = 1, 2, . . .

Programming Considerations:

For reasons of programming efficiency and for diminishing roundoff errors, the recurrence relations are modified to the following forms:

Chebyshev polynomials

$T_{-1} = x,\ T_0 = 1,\ T_{k+1} = xT_k - T_{k-1} + xT_k$

for k = 0, 1, 2, . . . , n-2

Legendre polynomials

$P_{-1} = 0,\ P_0 = 1,$

$P_{k+1} = xP_k - P_{k-1} - (xP_k - P_{k-1})/(k+1) + xP_k$

for k = 0, 1, 2, . . . , n - 2

Laguerre polynomials

$L_{-1} = 0,\ L_0 = 1,$

$L_{k+1} = L_k - L_{k-1} + (L_k - (xL_k - L_{k-1} + L_k)/(k+1))$

for k = 0, 1, 2, . . . , n - 2

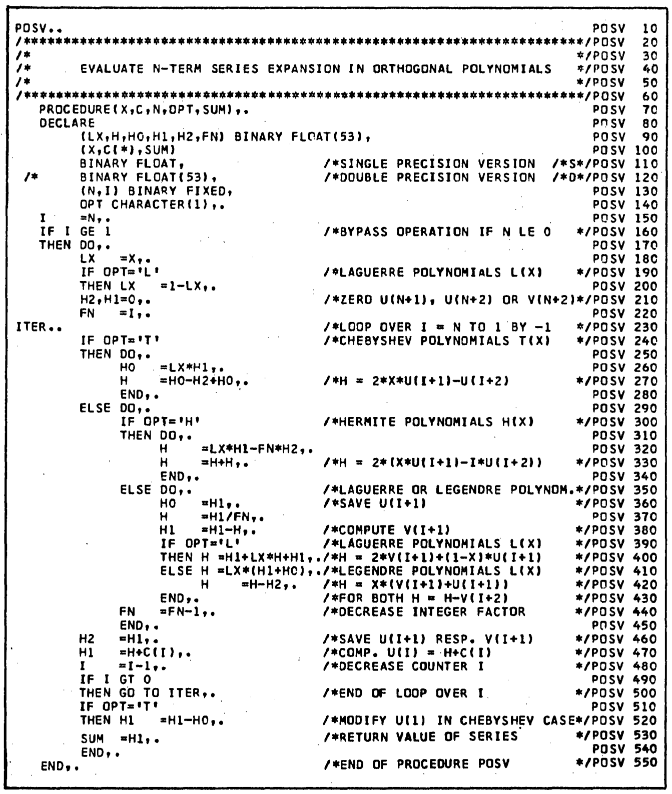
Hermite polynomials

$H_{-1} = 0,\ H_0 = 1,$

$H_{k+1} = xH_k - H_{k-1} - (2k-1)H_{k-1} + xH_k$

for k = 0, 1, 2, . . . , n-2

- Subroutine POSV



```
POSV..
/**********************************************************************/
/*                                                                    */
/*     EVALUATE N-TERM SERIES EXPANSION IN ORTHOGONAL POLYNOMIALS     */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,C,N,OPT,SUM),.
   DECLARE
        (LX,H,HO,H1,H2,FN) BINARY FLOAT(53),
        (X,C(*),SUM)
        BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),               /*DOUBLE PRECISION VERSION  /*D*/
        (N,I) BINARY FIXED,
        OPT CHARACTER(1),.
   I    =N,.
   IF I GE 1                            /*BYPASS OPERATION IF N LE 0   */
   THEN DO,.
        LX   =X,.
        IF OPT='L'                      /*LAGUERRE POLYNOMIALS L(X)    */
        THEN LX   =1-LX,.
        H2,H1=0,.                       /*ZERO U(N+1), U(N+2) OR V(N+2)*/
        FN   =I,.
ITER..                                  /*LOOP OVER I = N TO 1 BY -1   */
        IF OPT='T'                      /*CHEBYSHEV POLYNOMIALS T(X)   */
        THEN DO,.
             HO   =LX*H1,.
             H    =HO-H2+HO,.           /*H = 2*X*U(I+1)-U(I+2)        */
             END,.
        ELSE DO,.
             IF OPT='H'                 /*HERMITE POLYNOMIALS H(X)     */
             THEN DO,.
                  H    =LX*H1-FN*H2,.
                  H    =H+H,.           /*H = 2*(X*U(I+1)-I*U(I+2))    */
                  END,.
             ELSE DO,.                  /*LAGUERRE OR LEGENDRE POLYNOM.*/
                  HO   =H1,.            /*SAVE U(I+1)                  */
                  H    =H1/FN,.
                  H1   =H1-H,.          /*COMPUTE V(I+1)               */
                  IF OPT='L'            /*LAGUERRE POLYNOMIALS L(X)    */
                  THEN H =H1+LX*H+H1,./*H = 2*V(I+1)+(1-X)*U(I+1)      */
                  ELSE H =LX*(H1+HO),./*LEGENDRE POLYNOMIALS L(X)      */
                       H    =H-H2,.   /*H = X*(V(I+1)+U(I+1))          */
                  END,.                 /*FOR BOTH H = H-V(I+2)        */
             FN   =FN-1,.               /*DECREASE INTEGER FACTOR      */
             END,.
        H2   =H1,.                      /*SAVE U(I+1) RESP. V(I+1)     */
        H1   =H+C(I),.                  /*COMP. U(I) = H+C(I)          */
        I    =I-1,.                     /*DECREASE COUNTER I           */
        IF I GT 0
        THEN GO TO ITER,.               /*END OF LOOP OVER I           */
        IF OPT='T'
        THEN H1   =H1-HO,.              /*MODIFY U(1) IN CHEBYSHEV CASE*/
        SUM  =H1,.                      /*RETURN VALUE OF SERIES       */
        END,.
   END,.                                /*END OF PROCEDURE POSV        */
```

Purpose :

POSV computes the value of the sum

$\sum_{k=1}^{N} c_k f_{k-1}(x)$ for a given vector $C = (c_1, c_2, \ldots, c_N)$,

and a specified set of orthogonal polynomials (f_k).

The user has the choice of

Chebyshev polynomials $(T_0, T_1, \ldots, T_{N-1})$
with OPT = 'T'

Legendre polynomials $(P_0, P_1, \ldots, P_{N-1})$
with OPT = 'P'

Laguerre polynomials $(L_0, L_1, \ldots, L_{N-1})$
with OPT = 'L'

Hermite polynomials $(H_0, H_1, \ldots, H_{N-1})$
with OPT = 'H'

Usage:

CALL POSV (X, C, N, OPT, SUM) ;

X - BINARY FLOAT [(53)]
Given argument of orthogonal polynomials.

C(N) - BINARY FLOAT [(53)]
Given coefficient vector of series expansion.

N - BINARY FIXED
Given dimension of coefficient vector.

OPT - CHARACTER (1)
Given parameter of choice (see "Purpose").

SUM - BINARY FLOAT [(53)]
Resultant value of series expansion for argument X.

Remarks:

Operation is bypassed if N is not positive. Any input value of OPT other than 'T', 'L', or 'H' is treated as if it were 'P'.

The sum of an expansion in shifted Chebyshev or Legendre polynomials for argument x is obtained as the value of the expansion in non-shifted polynomials for argument (2 • x - 1).

Method:

Evaluation is based on the three-term recurrence relation for orthogonal polynomials, using a backward iteration scheme.

For reference see:

M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, Applied Mathematics Series 55, National Bureau of Standards, 1964, pp. 771-803.

Mathematical Background:

Evaluation is based on the following iteration schemes:

Chebyshev expansion

Set $U_{n+1} = U_{n+2} = 0$ and use the recurrence relations:

$$T_k = 2xT_{k-1} - T_{k-2}, \quad U_k = c_k + 2xU_{k+1} - U_{k+2}$$

successively for k = n, n-1, . . . , 2.

Then

$$\sum_{i=1}^{n} c_i T_{i-1} = \sum_{i=1}^{n} c_i T_{i-1} + U_{n+1} T_n - U_{n+2} \cdot T_{n-1}$$

$$= \sum_{i=1}^{n-1} c_i T_{i-1} + (c_n + 2xU_{n+1} - U_{n+2}) T_{n-1}$$

$$- U_{n+1} T_{n-2}$$

$$= \sum_{i=1}^{n-1} c_i T_{i-1} + U_n T_{n-1} - U_{n+1} T_{n-2}$$

$$\vdots$$

$$= c_1 T_0 + U_2 T_1 - U_3 T_0 = c_1 + xU_2 - U_3$$

Legendre expansion

Set $U_{n+1} = U_{n+2} = 0$ and use the recurrence relations

$$kP_k = x(2k-1)\,P_{k-1} - (k-1)P_{k-2}$$

$$(k-1)U_k = c_k + x(2k-1)U_{k+1} - kU_{k+2}$$

successively for k = n, n-1, . . . , 2. Then:

$$\sum_{i=1}^{n} c_i P_{i-1} = \sum_{i=1}^{n} c_i P_{i-1} + U_{n+1} \cdot nP_n - nU_{n+2} \cdot P_{n-1}$$

$$= \sum_{i=1}^{n-1} c_i P_{i-1} + (c_n + x(2n-1)U_{n+1} - nU_{n+2})\,P_{n-1}$$

$$- U_{n+1}(n-1)P_{n-2}$$

$$= \sum_{i=1}^{n-1} c_i P_{i-1} + U_n (n-1)P_{n-1} - (n-1)$$

$$U_{n+1} \cdot P_{n-2}$$

$$= c_1 P_0 + U_2 \cdot P_1 - U_3 P_0 = c_1 + xU_2 - U_3$$

Laguerre expansion

Set $U_{n+1} = U_{n+2} = 0$ and use the recurrence relations $kL_k = (2k-1-x)L_{k-1} - (k-1)L_{k-2}$

$(k-1)U_k = c_k + (2k-1-x)U_{k+1} - kU_{k+2}$

sucessively for k = n, n-1, . . . , 2. Then:

$$\begin{aligned}
\sum_{i=1}^{n} c_i L_{i-1} &= \sum_{i=1}^{n} c_i L_{i-1} + U_{n+1} \cdot nL_n - nU_{n+2} L_{n-1} \\
&= \sum_{i=1}^{n-1} c_i L_{i-1} + (c_n + (2n-1-x)U_{n+1} \\
&\quad -nU_{n+2}) L_{n-1} - (n-1)U_{n+1} L_{n-2} \\
&= \sum_{i=1}^{n-1} c_i L_{i-1} + U_n \cdot (n-1)L_{n-1} \\
&\quad - (n-1) U_{n+1} L_{n-2} \\
&\quad \vdots \\
&= c_1 L_0 + U_2 L_1 - U_3 L_0 \\
&= c_1 + U_2(1-x) - U_3
\end{aligned}$$

Hermite Expansion

Set $U_{n+1} = U_{n+2} = 0$ and use the recurrence relations $H_k = 2xH_{k-1} - 2(k-1)H_{k-2}$

$U_k = c_k + 2xU_{k+1} - 2kU_{k+2}$

successively for k = n, n-1, ..., 2. Then:

$$\begin{aligned}
\sum_{i=1}^{n} c_i H_{i-1} &= \sum_{i=1}^{n} c_i H_{i-1} + U_{n+1} H_n - 2nU_{n+2} \cdot H_{n-1} \\
&= \sum_{i=1}^{n-1} c_i H_{i-1} + (C_n + 2xU_{n+1} - 2nU_{n+2}) H_{n-1} \\
&\quad - 2(n-1)U_{n+1} \cdot H_{n-2} \\
&= \sum_{i=1}^{n-1} c_i H_{i-1} + U_n \cdot H_{n-1} \\
&\quad -2(n-1) U_{n+1} H_{n-2} \\
&\quad \vdots \\
&= c_1 H_0 + U_2 \cdot H_1 - 2U_3 H_0 \\
&= c_1 + 2xU_2 - 2U_3
\end{aligned}$$

Programming Considerations:

For reasons of programming efficiency the following modifications of the backward iteration scheme are used for evaluations:

Chebyshev expansion

Set:

$$U_{n+1} = U_{n+2} = 0$$

$$U_i = xU_{i+1} - U_{i+2} + xU_{i+1} + c_i \text{ for } i = n, \ldots, 1$$

Then:

$$\sum_{i=1}^{n} c_i T_{i-1}(x) = U_1 - xU_2$$

Legendre expansion

Set:

$$U_{n+1} = V_{n+2} = 0$$

$$\left.\begin{aligned} V_{i+1} &= U_{i+1} - U_{i+1/i} \\ U_i &= x(V_{i+1} - U_{i+1}) - V_{i+2} \end{aligned}\right\} \text{ for } i = n, \ldots, 1$$

Then:

$$\sum_{i=1}^{n} c_i P_{i-1}(x) = U_1$$

Laguerre expansion

Set:

$$U_{n+1} = V_{n+2} = 0$$

$$\left.\begin{aligned} V_{i+1} &= U_{i+1} - U_{i+1/i} \\ U_i &= V_{i+1} + (1-x) U_{i+1/i} + V_{i+1} - V_{i+2} + c_i \end{aligned}\right\}$$

for i = n, . . . , 1

Then:

$$\sum_{i=1}^{n} c_i L_{i-1}(x) = U_1$$

Hermite expansion

Set:

$$U_{n+1} = U_{n+2} = 0$$

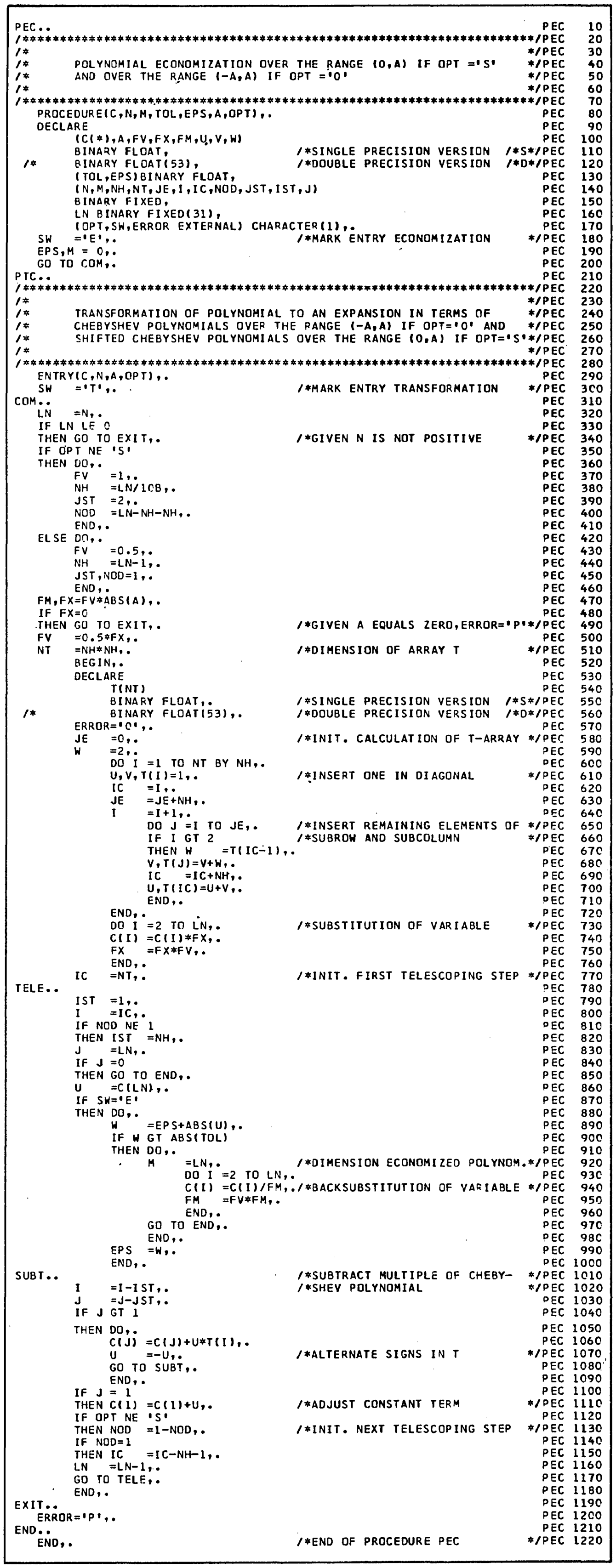
$$U_i = (xU_{i+1} - i \cdot U_{i+2}) + (xU_{i+1} - iU_{i+2})$$

for $i = n, \ldots, 1$

Then:

$$\sum_{i=1}^{n} c_i H_{i-1}(x) = U_1$$

- Subroutine PEC/PTC



```
PEC..
/**********************************************************************/
/*                                                                    */
/*      POLYNOMIAL ECONOMIZATION OVER THE RANGE (O,A) IF OPT ='S'     */
/*      AND OVER THE RANGE (-A,A) IF OPT ='O'                         */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(C,N,M,TOL,EPS,A,OPT),.
   DECLARE
        (C(*),A,FV,FX,FM,U,V,W)
        BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),               /*DOUBLE PRECISION VERSION  /*D*/
        (TOL,EPS)BINARY FLOAT,
        (N,M,NH,NT,JE,I,IC,NOD,JST,IST,J)
        BINARY FIXED,
        LN BINARY FIXED(31),
        (OPT,SW,ERROR EXTERNAL) CHARACTER(1),.
   SW   ='E',.                          /*MARK ENTRY ECONOMIZATION    */
   EPS,M = 0,.
   GO TO COM,.
PTC..
/**********************************************************************/
/*                                                                    */
/*      TRANSFORMATION OF POLYNOMIAL TO AN EXPANSION IN TERMS OF      */
/*      CHEBYSHEV POLYNOMIALS OVER THE RANGE (-A,A) IF OPT='O' AND    */
/*      SHIFTED CHEBYSHEV POLYNOMIALS OVER THE RANGE (O,A) IF OPT='S'*/
/*                                                                    */
/**********************************************************************/
   ENTRY(C,N,A,OPT),.
   SW   ='T',.                          /*MARK ENTRY TRANSFORMATION   */
COM..
   LN   =N,.
   IF LN LE 0
   THEN GO TO EXIT,.                    /*GIVEN N IS NOT POSITIVE     */
   IF OPT NE 'S'
   THEN DO,.
        FV   =1,.
        NH   =LN/10B,.
        JST  =2,.
        NOD  =LN-NH-NH,.
        END,.
   ELSE DO,.
        FV   =0.5,.
        NH   =LN-1,.
        JST,NOD=1,.
        END,.
   FM,FX=FV*ABS(A),.
   IF FX=0
   THEN GO TO EXIT,.                    /*GIVEN A EQUALS ZERO,ERROR='P'*/
   FV   =0.5*FX,.
   NT   =NH*NH,.                        /*DIMENSION OF ARRAY T        */
        BEGIN,.
        DECLARE
             T(NT)
             BINARY FLOAT,.             /*SINGLE PRECISION VERSION  /*S*/
 /*          BINARY FLOAT(53),.         /*DOUBLE PRECISION VERSION  /*D*/
        ERROR='0',.
        JE   =0,.                       /*INIT. CALCULATION OF T-ARRAY */
        W    =2,.
             DO I =1 TO NT BY NH,.
             U,V,T(I)=1,.               /*INSERT ONE IN DIAGONAL      */
             IC   =I,.
             JE   =JE+NH,.
             I    =I+1,.
                  DO J =I TO JE,.       /*INSERT REMAINING ELEMENTS OF */
                  IF I GT 2             /*SUBROW AND SUBCOLUMN        */
                  THEN W    =T(IC-1),.
                  V,T(J)=V+W,.
                  IC   =IC+NH,.
                  U,T(IC)=U+V,.
                  END,.
             END,.
             DO I =2 TO LN,.            /*SUBSTITUTION OF VARIABLE    */
             C(I) =C(I)*FX,.
             FX   =FX*FV,.
             END,.
        IC   =NT,.                      /*INIT. FIRST TELESCOPING STEP */
TELE..
        IST  =1,.
        I    =IC,.
        IF NOD NE 1
        THEN IST  =NH,.
        J    =LN,.
        IF J =0
        THEN GO TO END,.
        U    =C(LN),.
        IF SW='E'
        THEN DO,.
             W    =EPS+ABS(U),.
             IF W GT ABS(TOL)
             THEN DO,.
                  M    =LN,.            /*DIMENSION ECONOMIZED POLYNOM.*/
                       DO I =2 TO LN,.
                       C(I) =C(I)/FM,./*BACKSUBSTITUTION OF VARIABLE */
                       FM   =FV*FM,.
                       END,.
                  GO TO END,.
                  END,.
             EPS  =W,.
             END,.
SUBT..                                  /*SUBTRACT MULTIPLE OF CHEBY-  */
        I    =I-IST,.                   /*SHEV POLYNOMIAL             */
        J    =J-JST,.
        IF J GT 1
        THEN DO,.
             C(J) =C(J)+U*T(I),.
             U    =-U,.                 /*ALTERNATE SIGNS IN T        */
             GO TO SUBT,.
             END,.
        IF J = 1
        THEN C(1) =C(1)+U,.             /*ADJUST CONSTANT TERM        */
        IF OPT NE 'S'
        THEN NOD  =1-NOD,.              /*INIT. NEXT TELESCOPING STEP  */
        IF NOD=1
        THEN IC   =IC-NH-1,.
        LN   =LN-1,.
        GO TO TELE,.
        END,.
EXIT..
   ERROR='P',.
END..
   END,.                                /*END OF PROCEDURE PEC        */
```

Purpose:

PEC approximates a given polynomial by a polynomial of lower degree, using a telescoping technique, so that the error does not exceed a user-specified tolerance TOL. Range of approximation is (-a, a) if OPT='0' and (0, a) if OPT='S'.

Usage:

CALL PEC (C, N, M, TOL, EPS, A, OPT);

C(N) - BINARY FLOAT [(53)]
Given coefficient vector of the polynomial $P(x) = c_1 + c_2x + ... +c_nx^{n-1}$
Resultant coefficient vector of the economized polynomial $P_{m-1}(x) = c_1 + c_2x + ... + c_mx^{m-1}$

N - BINARY FIXED
Given dimension of given coefficient vector.

M - BINARY FIXED
Resultant dimension of economized coefficient vector.

TOL - BINARY FLOAT
Given tolerance specified by the user.

EPS - BINARY FLOAT
Resultant bound for the absolute difference between the given and economized polynomial over the specified range.

A - BINARY FLOAT [(53)]
Given value defining the range of approximation.

OPT - CHARACTER(1)
Given option for selection of operation

Purpose:

PTC transforms a given polynomial into an expansion of Chebyshev polynomials if OPT = '0' and of shifted Chebyshev polynomials if OPT = 'S'.

Usage:

CALL PTC (C, N, A, OPT);

C(N) - BINARY FLOAT [(53)]
Given coefficient vector of the polynomial $P(x) = c_1 + c_2x + ... + c_nx^{n-1}$
Resultant coefficient vector of Chebyshev expansion
$P(x) = c_1 + c_2t_1(t) + ... + c_nt_{n-1}(t)$
with $t = x/A$

$$\text{and } t_k(t) = \begin{cases} T_k(t) & \text{if OPT='0'} \\ T^*_k(t) & \text{if OPT='S'} \end{cases}$$

N - BINARY FIXED
Given dimension of the coefficient vector.

A - BINARY FLOAT [(53)]
Given value defining the range of expansion.

OPT - CHARACTER (1)
Given option for selection of operation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR = 'P' means invalid parameters:
either $N \le 0$ or $A = 0$

A value of OPT different from 'S' is interpreted as if it were '0'.

On return from PEC the locations $c_{m+1}, ..., c_n$ contain the coefficients of the Chebyshev expansion of the difference between the given polynomial P(x) and the economized polynomial $P_{m-1}(x)$:

$$P(x) = P_{m-1}(x) + c_{m+1}t_m(t) + ... c_nt_{n-1}(t)$$

Therefore, using PEC with a very large tolerance TOL (say, 10^{75}) has the same effect as the application of PTC.

Method:

In the first telescoping step a multiple of the Chebyshev polynomial

$$t_{n-1}(x/a) = \begin{cases} T_{n-1}(x/a) & \text{if OPT = '0'} \\ T^*_{n-1}(x/a) & \text{if OPT = 'S'} \end{cases}$$

is subtracted from given P(x), so that the difference is a polynomial of degree n-2.
Set:

$$P_{n-1}(x) = P(x)$$

then:

$$P_{n-2}(x) = P_{n-1}(x) - b_nt_{n-1}(x/a) \qquad (1)$$

Telescoping $P_{n-2}(x)$ again results in a polynomial $P_{n-3}(x)$ of degree n-3, and by iteration

$$P(x) = b_1 + b_2t_1(x/a) + b_3t_2(x/a) + ... + b_nt_{n-1}(x/a) \qquad (2)$$

This means that calculated b's form the coefficient vector of the expansion in terms of Chebyshev polynomials. If telescoping steps are performed only as long as

$$\left| b_n \right| + \left| b_{n-1} \right| + \dots + \left| b_{m+1} \right| \leq \left| \text{TOL} \right.$$

then $P_{m-1}(x)$ is the economized polynomial. For the Chebyshev polynomials

$$\left| t_k(x/a) \right| \leq 1 \text{ for } \left| x \right| \leq a$$

and for all values of k; therefore,

$$\begin{aligned} \left| P(x) - P_{m-1}(x) \right| &= \left| b_{m+1} t_m (x/a + \dots \right. \\ &\quad \left. + b_n t_{n-1}(x/a) \right| \\ &\leq \left| b_{m+1} \right| + \left| b_{m+2} \right| \\ &\quad + \dots + \left| b_n \right| \leq \left| \text{TOL} \right| \end{aligned} \tag{3}$$

Mathematical Background

Calculation of the coefficients of $T_k(t)$

$$\text{Set } C_k(z) = 2T_k\,(z/2) \text{ or } T_k(t) = \frac{1}{2} C_k(2t), \text{ with } t = \frac{z}{2}. \tag{4}$$

$$\text{Then } C_k\,(z) = S_k(z) - S_{k-2}(z) \tag{5}$$

$$\text{with } S_k(z) = \binom{k}{0} z^k - \binom{k-1}{1} z^{k-1} + - \dots \pm \binom{0}{k}. \tag{6}$$

The binomial coefficients $\binom{k-v}{v}$ are easily generated using Pascal's triangle.

An analogous calculation scheme exists for the coefficients of $C_k\,(z)$:

$$\begin{aligned} C_k(z) &= \frac{k}{k}\binom{k}{0} z^k - \frac{k}{k-1}\binom{k-1}{1} z^{k-z} \\ &\quad + \frac{k}{k-2}\binom{k-2}{2} z^{k-4} - \dots \end{aligned} \tag{7}$$

The coefficients of successive $C_k\,(z)$ are easily found by the calculation scheme

$$\begin{array}{llllllll}
2 & & & & & C_0(z) = 2 & & \\
1 & & & & & & & C_1(z) = z \\
2 & 1 & & & & C_2(z) = z^2 - 2 & & \\
3 & 1 & & & & & & C_3(z) = z^2 - 3z \\
2 & 4 & 1 & & & C_4(z) = z^4 - 4z^2 + 2 & & \\
5 & \textcircled{5} & 1 & & & & & C_5(z) = z^5 - 5z^3 + 5z \\
2 & \textcircled{9} & 6 & 1 & & C_6(z) = z^6 - 6z^4 + 9z^2 - 2 & & \\
7 & \textcircled{14} & 7 & 1 & & . & & . \\
2 & 16 & 20 & 8 & 1 & . & & . \\
. & . & . & . & . & . & & . \\
. & . & . & . & . & . & & .
\end{array} \tag{8}$$

The above calculation scheme means that the first column is all two's and the diagonal elements are all ones. The remaining elements are obtained by adding the two elements above in the same column and in the adjacent left-hand column. For example, circled element 14 is obtained by adding the two circled elements 9 and 5.

The shifted Chebyshev polynomials are reduced to ordinary ones using the identity

$$2T_k^*\;(u/4) = 2T_{2k}\,(\sqrt{u}/2) = C_{2k}\,(\sqrt{u}) \tag{9}$$

or

$$T_k^*\;(t) = \frac{1}{2} C_{2k}\,(2\sqrt{t}) \quad \text{with } t = u/4$$

Programming Considerations:

The triangle (8) may be stored more compactly in the rectangular scheme:

$$\begin{array}{cccccc}
2 & 1 & 3 & 5 & 7 & . \\
2 & 4 & 1 & 5 & 14 & . \\
2 & 9 & 6 & 1 & 7 & . \\
2 & 16 & 20 & 8 & 1 & . \\
. & . & . & . & . & .
\end{array} \tag{10}$$

The coefficients of C_{2k-1} form subcolumns and those of C_{2k} corresponding subrows. In order to be able to use the coefficients of the auxiliary array (10), the given polynomial

$$P(x) = c_1 + c_2x + \dots + c_n x^{n-1} \qquad (11)$$

must first be transformed substituting $x = |a|\, t$, which gives

$$P(x) = b_1 + b_2 t + b_3 t^2 + \dots + b_n t^{n-1} \qquad (12)$$

By this the argument range gets reduced to the standard interval (-1, +1) if OPT = '0' and (0, 1) if OPT = 'S'.

The next step is to introduce z=2t if OPT = '0' and u=4t if OPT = 'S' and to divide all coefficients so obtained by two, except the first one. Naturally the two substitutions may be applied simultaneously:

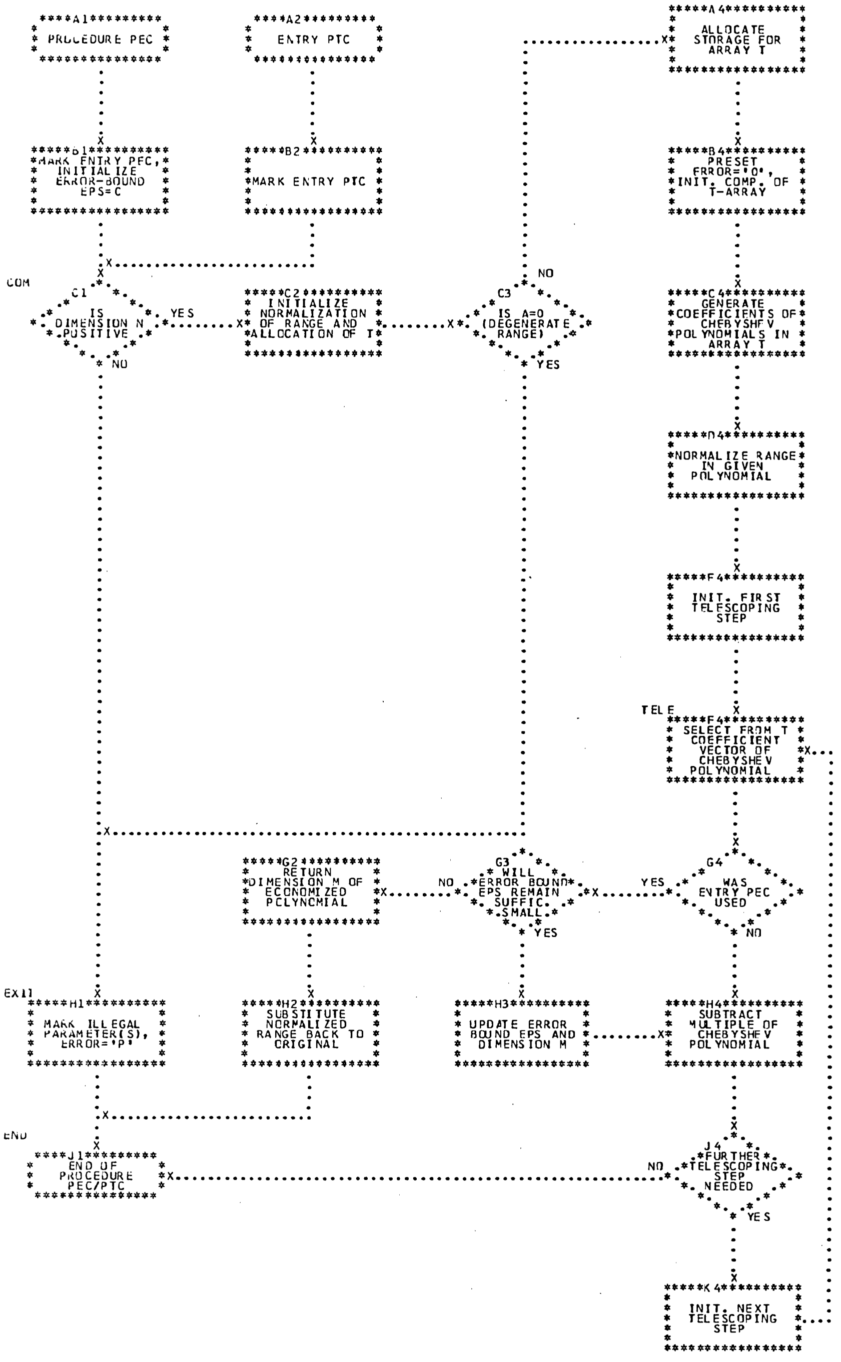
$$x = |a| \cdot t = \frac{|a|}{2} z = \frac{|a|}{4} u \qquad (13)$$

The sequence of calculations performed is as follows:

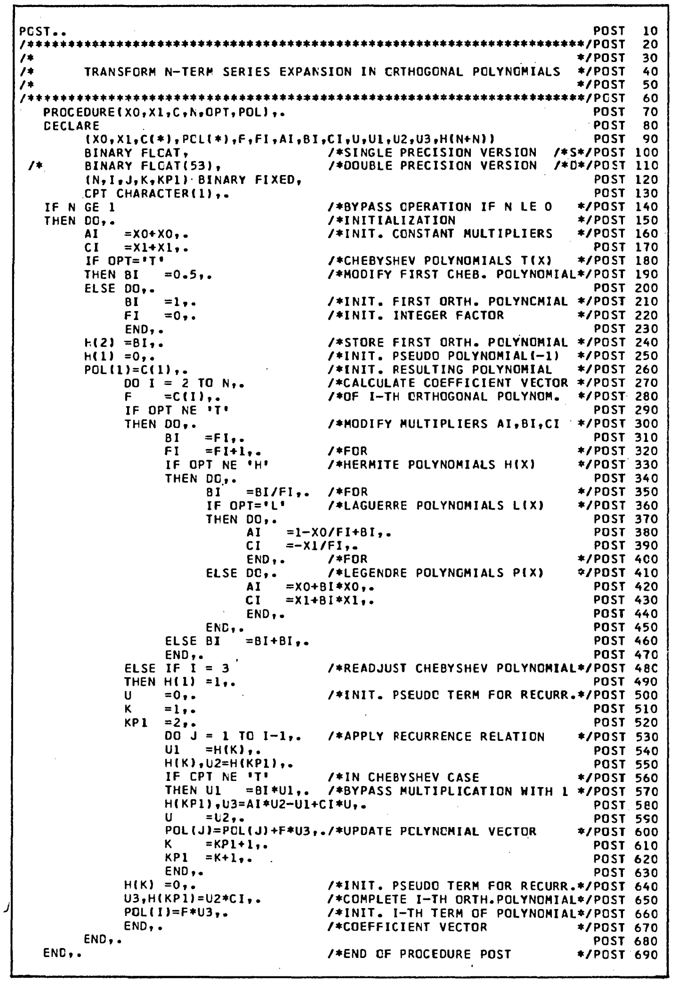
1. The auxiliary array (10) is set up calculating row and column simultaneously.
2. The given coefficient vector gets replaced by the coefficient vector with variable z or u.
3. In case PTC, performing n-1 successive telescoping steps gives the expansion in terms of Chebyshev polynomials. In case PEC, the iterative telescoping is stopped as soon as the tolerance TOL is exceeded.
4. The economized polynomial must be back-transformed to the original variable x.

PROCEDURE PEC ECONOMIZES A POLYNOMIAL USING TRUNCATION IN CORRESPONDING CHEBYSHEV EXPANSION

ENTRY PTC TRANSFORMS A POLYNOMIAL TC AN EXPANSION IN TERMS OF CHEBYSHEV POLYNOMIALS


A1
PROCEDURE PEC
A2
ENTRY PTC
A4
ALLOCATE STORAGE FOR ARRAY T
B1
MARK ENTRY PEC, INITIALIZE ERROR-BOUND EPS=C
B2
MARK ENTRY PTC
B4
PRESET ERROR='0', INIT. COMP. OF T-ARRAY
COM
C1
IS DIMENSION N POSITIVE
YES
NO
C2
INITIALIZE NORMALIZATION OF RANGE AND ALLOCATION OF T
C3
IS A=0 (DEGENERATE RANGE)
NO
YES
C4
GENERATE COEFFICIENTS OF CHEBYSHEV POLYNOMIALS IN ARRAY T
D4
NORMALIZE RANGE IN GIVEN POLYNOMIAL
E4
INIT. FIRST TELESCOPING STEP
TELE
F4
SELECT FROM T COEFFICIENT VECTOR OF CHEBYSHEV POLYNOMIAL
G2
RETURN DIMENSION M OF ECONOMIZED POLYNOMIAL
G3
WILL ERROR BOUND EPS REMAIN SUFFIC. SMALL
NO
YES
G4
WAS ENTRY PEC USED
YES
NO
EXIT
H1
MARK ILLEGAL PARAMETER(S), ERROR='P'
H2
SUBSTITUTE NORMALIZED RANGE BACK TO ORIGINAL
H3
UPDATE ERROR BOUND EPS AND DIMENSION M
H4
SUBTRACT MULTIPLE OF CHEBYSHEV POLYNOMIAL
END
J1
END OF PROCEDURE PEC/PTC
J4
FURTHER TELESCOPING STEP NEEDED
NO
YES
K4
INIT. NEXT TELESCOPING STEP

- <u>Subroutine</u> POST



```
PCST..
/*********************************************************************/
/*                                                                   */
/*      TRANSFORM N-TERM SERIES EXPANSION IN CRTHOGONAL POLYNOMIALS  */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(X0,X1,C,N,OPT,POL),.
   CECLARE
        (X0,X1,C(*),PCL(*),F,FI,AI,BI,CI,U,U1,U2,U3,H(N+N))
        BINARY FLCAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLCAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (N,I,J,K,KP1) BINARY FIXED,
        CPT CHARACTER(1),.
   IF N GE 1                          /*BYPASS OPERATION IF N LE 0   */
   THEN DO,.                          /*INITIALIZATION               */
        AI   =X0+X0,.                 /*INIT. CONSTANT MULTIPLIERS   */
        CI   =X1+X1,.
        IF OPT='T'                    /*CHEBYSHEV POLYNOMIALS T(X)   */
        THEN BI   =0.5,.              /*MODIFY FIRST CHEB. POLYNOMIAL*/
        ELSE DO,.
             BI   =1,.                /*INIT. FIRST ORTH. POLYNCMIAL */
             FI   =0,.                /*INIT. INTEGER FACTOR         */
             END,.
        H(2) =BI,.                    /*STORE FIRST ORTH. POLYNOMIAL */
        H(1) =0,.                     /*INIT. PSEUDO POLYNOMIAL(-1)  */
        POL(1)=C(1),.                 /*INIT. RESULTING POLYNOMIAL   */
             DO I = 2 TO N,.          /*CALCULATE COEFFICIENT VECTOR */
             F    =C(I),.             /*OF I-TH CRTHOGONAL POLYNOM.  */
             IF OPT NE 'T'
             THEN DO,.                /*MODIFY MULTIPLIERS AI,BI,CI  */
                  BI   =FI,.
                  FI   =FI+1,.        /*FOR                          */
                  IF OPT NE 'H'       /*HERMITE POLYNOMIALS H(X)     */
                  THEN DO,.
                       BI   =BI/FI,.  /*FOR                          */
                       IF OPT='L'     /*LAGUERRE POLYNOMIALS L(X)    */
                       THEN DO,.
                            AI   =1-X0/FI+BI,.
                            CI   =-X1/FI,.
                            END,.     /*FOR                          */
                       ELSE DC,.      /*LEGENDRE POLYNCMIALS P(X)    */
                            AI   =X0+BI*X0,.
                            CI   =X1+BI*X1,.
                            END,.
                       END,.
                  ELSE BI   =BI+BI,.
                  END,.
             ELSE IF I = 3            /*READJUST CHEBYSHEV POLYNOMIAL*/
             THEN H(1) =1,.
             U    =0,.                /*INIT. PSEUDC TERM FOR RECURR.*/
             K    =1,.
             KP1  =2,.
                  DO J = 1 TO I-1,.   /*APPLY RECURRENCE RELATION    */
                  U1   =H(K),.
                  H(K),U2=H(KP1),.
                  IF CPT NE 'T'       /*IN CHEBYSHEV CASE            */
                  THEN U1   =BI*U1,.  /*BYPASS MULTIPLICATION WITH 1 */
                  H(KP1),U3=AI*U2-U1+CI*U,.
                  U    =U2,.
                  POL(J)=PCL(J)+F*U3,./*UPDATE PCLYNCMIAL VECTOR     */
                  K    =KP1+1,.
                  KP1  =K+1,.
                  END,.
             H(K) =0,.                /*INIT. PSEUDO TERM FOR RECURR.*/
             U3,H(KP1)=U2*CI,.        /*COMPLETE I-TH ORTH.POLYNOMIAL*/
             POL(I)=F*U3,.            /*INIT. I-TH TERM OF POLYNOMIAL*/
             END,.                    /*COEFFICIENT VECTOR           */
        END,.
   END,.                              /*END CF PROCEDURE POST        */
```

Purpose:

POST transforms a given series expansion in orthogonal polynomials to a polynomial. The independent variable of the given expansion is assumed to be $x_0 + x_1 x$; that is, a linear transformation of the range is built in. The coefficient vector $C = (c_1, \ldots, c_n)$ is given. Procedure POST calculates $POL = (pol_1, \ldots, pol_n)$ satisfying

$$\sum_{i=1}^{n} c_i f_{i-1} (x_0 + x_1 \cdot x) = \sum_{i=1}^{n} pol_i \cdot x^{i-1}$$

For the specified set of orthogonal polynomials (f_k) the user has the choice of:

Chebyshev polynomials $(T_0, T_1, \ldots, T_{n-1})$
with OPT = 'T'
Legendre polynomials $(P_0, P_1, \ldots, P_{n-1})$
with OPT = 'P'
Laguerre polynomials $(L_0, L_1, \ldots, L_{n-1})$
with OPT = 'L'
Hermite polynomials $(H_0, H_1, \ldots, H_{n-1})$
with OPT = 'H'

Usage:

CALL POST (X0, X1, C, N, OPT, POL);

X0 - BINARY FLOAT [(53)]
Given constant term of argument transformation.
X1 - BINARY FLOAT [(53)]
Given linear term of argument transformation.
C(N) - BINARY FLOAT [(53)]
Given coefficient vector of expansion, with coefficients ordered from low to high.
N - BINARY FIXED
Given dimension of coefficient vector.
OPT - CHARACTER (1)
Given parameter of choice (see "purpose").
POL(N) - BINARY FLOAT [(53)]
Resultant coefficient vector of resultant ordinary polynomial, with coefficients ordered from low to high.

Remarks:

N must be positive, or operation is bypassed.

Any input value of OPT other than 'T', 'L', or 'H' is treated as if it were 'P'.

Transformation of an expansion in shifted Chebyshev or Legendre polynomials is obtained using the linear transformation $(2x_0 - 1) + (2x_1) x$.

The resultant vector POL may occupy the same storage locations as the given vector C.

Method:

The coefficient vector POL is calculated from the coefficient vectors of the orthogonal polynomials, which are generated successively using the recurrence relation.

$$f_{k+1} = (a_k + c_k x) f_k - b_k f_{k-1} \text{ for } k \geq 0$$

with $f_{-1}=0$, $f_0=1$.

For reference see:

M. Abramowitz/I. A. Stegun, <u>Handbook of Mathematical Functions</u>, Applied Mathematics Series 55, National Bureau of Standards, 1964, pp. 771-803.

Mathematical Background:

The coefficient vectors of the orthogonal polynomials for argument $z = x_0 + x_1 x$ are generated using the three-term recurrence relation:

Chebyshev polynomials

$$T_{-1} = 0,\ T_0 = 1,\ T_1(z) = x_0 + x_1 x$$

$$T_{k+1}(z) = 2x_0 T_k(z) - T_{k-1}(z) + 2x_1 \cdot xT_k(z),$$

for k ≥ 1

Legendre polynomials

$$P_{-1} = 0,\ P_0 = 1$$

$$P_{k+1}(z) = \left(1 + \frac{k}{k+1}\right) x_0 P_k(z) - \left(\frac{k}{k+1}\right) P_{k-1}(z)$$

$$+ \left(1 + \frac{k}{k+1}\right) x_1 x\ P_k(z),\ \text{for } k \geq 0$$

Laguerre polynomials

$$L_{-1} = 0,\ L_0 = 1$$

$$L_{k+1}(z) = \left(1 + \frac{k}{k+1} - \frac{x_0}{k+1}\right) L_k(z) - \left(\frac{k}{k+1}\right) L_{k-1}(z)$$

$$- \left(\frac{x_1}{k+1}\right) xL_k(z),\ \text{for } k \geq 0$$

Hermite polynomials

$$H_{-1} = 0,\ H_0 = 1$$

$$H_{k+1} = 2x_0\ H_k(z) - 2kH_{k-1}(z) + 2x_1\ xH_k(z),$$

for k ≥ 0

Programming Considerations:

Using $T_0/2$ instead of T_0, the above recurrence relation for Chebyshev polynomials is also valid for calculation of the coefficient vector of $T_1(z)$ with k = 0. The coefficient vectors of two successive orthogonal polynomials are combined in an auxiliary linear array H with coefficients of the lower polynomial in H(1), H(3), ..., and those of the higher polynomial in H(2), H(4), ...

Both coefficient vectors are ordered from low to high.

- Subroutine PRTC

```
(NOUNDERFLOW)..PRTC..
/**********************************************************************/
/*                                                                    */
/*      CALCULATE ALL RCCTS CF A COMPLEX PCLYNCMIAL                   */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(C,N),.
   CECLARE
        C(*) COMPLEX
        BINARY FLCAT,                    /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),                /*DOUBLE PRECISION VERSION  /*D*/
        (D(N),B(N),Z,DZ,V,W,U,ZC) COMPLEX
        BINARY FLOAT,                    /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),                /*DOUBLE PRECISION VERSION  /*D*/
        (N,LN,I,K,KD,J,JE)
        BINARY FIXED,
        (I1,IN DEFINED R,ID DEFINED AW,IR,IR1,IR2)
        BINARY FIXED(31),
        (AV,AVO,TCL,AZ,AW,R,RD,RKM,ARG,ARGV)
        BINARY FLOAT,                    /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),                /*DOUBLE PRECISION VERSION  /*D*/
        ERROR EXTERNAL CHARACTER(1),.
   I1   =1091567616,.
   LN   =N,.                             /*NUMBER CF MISSING ROOTS      */
   Z    =C,.
   ERRCR='0',.
ZERO..
   AVO  =1E75,.                          /*FORCE SHIFT CF ORIGIN        */
   IF LN LE 0
   THEN GO TO EXIT,.                     /*ALL RCCTS CALCULATED         */
   IF C(LN)=0
   THEN DO,.                             /*EXTRACT ZERO ROOT            */
        LN   =LN-1,.
        GO TO ZERC,.
        END,.
   CZ,Z =CONJG(Z),.
        DO I = 1 TO LN,.
        D(I),B(I)=C(I),.                 /*MOVE CCEFFICIEN, VECTOR      */
        END,.
VALUE..
   TOL  =0.2,.                           /*INIT. RCUND CFF BOUND        */
   AZ   =ABS(Z),.
   V    =1,.
        DO I = 1 TO LN,.                 /*CCMP. RCUND-CFF BCUND        */
        W    =D(I),.                     /*AND PCLYNCMIAL VALUE         */
        V,C(I)=W+V*Z,.
        TOL  =ABS(W)+AZ*TCL,.
        END,.
   TOL  =(TOL+4*(TOL-ABS(W)))
        *1.0E-6,..,.                     /*SINGLE PRECISION VERSION  /*S*/
 /*     *0.25E-15,.                      /*DOUBLE PRECISION VERSION  /*D*/
   AV   =ABS(V),.
   IF AV= 0 THEN GO TC RCOT,.
   IF AV LE TOL
   THEN IF AV GT AVO
   THEN DO,.                             /*STORE CALCULATED ROOT        */
ROOT..
        C(LN)=Z,.
        LN   =LN-1,.
        GC TO ZERO,.
        END,.
   ARGV =ATAN(-IMAG(V),-REAL(V)),.
   IF AV LT AVO                          /*HAS VALUE DECREASED          */
   THEN DO,.
        R    =AV,.
        RD,U =1,.
        IR   =(IN-I1)/LN,.
        KD,JE=LN,.
SHIFT..
        W    =1,.
             DO J=1 TO JE,.              /*SHIFT CF ORIGIN              */
             B(J),W=B(J)+W*DZ,.
             END,.
        IF LN NE JE
        THEN DO,.
             AW   =ABS(W),.
             K    =LN-JE,.
             IR1  =(IN-ID)/K,.
             IF IR1 LT IR
             THEN DO,.
                  IR   =IR1,.
                  RD   =AW,.
                  U    =W,.
                  KD   =K,.
                  END,.
             END,.
        JE   =JE-1,.
        IF JE GE 1
        THEN GO TC SHIFT,.
        RKM  =1/FLOAT(KC),.
        R    =(AV/RD)**RKM,.
        ARG  =(ARGV-ATAN(IMAG(U),REAL(U)))*RKM,.
        ZO   =Z,.
        AVO  =AV,.
INCR..
        REAL(DZ)=R*CCS(ARG),.
        IMAG(DZ)=R*SIN(ARG),.
        Z    =ZO+DZ,.
        IF ZO NE Z
        THEN GO TO VALUE,.
        IF AV GT TOL
        THEN ERROR='C',.
        GO TO ROOT,.
        END,.
    ELSE CO,.                            /*MODIFY STEPSIZE TO DECREASE  */
        R    =R/2,.                      /*PCLYNCMIAL VALUE             */
        IR2  =(IN-I1)/1C000C0000B,.
        KD   =LN,.
        U    =1,.
        IR   =I1/10CCC0CCOCB,.
        K    =0,.
             DO J = LN-1 TO 1 BY -1,.
             K    =K+1,.
             W    =B(J),.
             AW   =ABS(W),.
             IR1  =ID/100C0C0000B-(LN-K)*IR2,.
             IF IR LT IR1
             THEN DC,.
                  KD   =K,.
                  U    =W,.
                  IR   =IR1,.
                  END,.
             END,.
        ARG  =(ARGV-ATAN(IMAG(U),REAL(U)))/FLOAT(KD),.
        GC TO INCR,.
        END,.
EXIT..
   END,.                                /*END CF PROCEDURE PRTC       */
```

Purpose:

PRTC calculates all roots of a given complex polynomial.

Usage:

CALL PRTC (C, N);

C(N) - COMPLEX BINARY FLOAT L(53)]
Given coefficient vector of normalized polynomial

$$P(Z)=Z^N+C_1Z^{N-1}+\dots+C_N$$

Resultant N complex roots of given polynomial.

N - BINARY FIXED
Given dimension of coefficient vector.
N is also the degree of the polynomial and the number of roots to be calculated.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected.

ERROR='C' means that calculated roots are possibly inaccurate. The polynomial must be given in normalized form -- that is, the coefficient of Z^N should be one (and is not stored). The coefficient vector is replaced by the calculated roots, beginning with C(N). The coefficient vector must be complex. In the real polynomial case, the imaginary part of the coefficients must be set to zero before using PRTC. PRTC will compile with error message IEM 11051. However, the generated object code executes correctly.

Method:

The method used was proposed by K. Nickel. It is a generalization of Newton's method and is not sensitive to multiple roots.

For reference see:

K. Nickel, "Die numerische Berechnung der Wurzeln eines Polynoms", Numerische Mathematik, vol. 9 (1966), pp. 80-98.

K. Nickel, "Die Nullstellen eines Polynoms", Algorithmus 5, Computing, Vol. 2 (1967), iss. 3, pp. 284-290.

Mathematical Background:

Generalized Newton step

Let z_i be an approximation to a root of

$$P(z) = z^n + c_1 z^{n-1} + \dots + c_n \tag{1}$$

The next approximation is calculated from the coefficients of the shifted polynomial:

$$P(z) = b_0 (z - z_i)^n + b_i (z - z_i)^{n-1} + \dots + b_n \text{ with } b_0 = 1 \tag{2}$$

$$z_{i+1} = z_i + \sqrt[(n-k)]{\frac{-b_n}{b_k}} \tag{3}$$

where k is chosen so that

$$r_k = \underset{j=0,1,\dots n-1}{\mathrm{MIN}} \sqrt[(n-j)]{\left|\frac{b_n}{b_j}\right|} = \sqrt[(n-k)]{\left|\frac{b_n}{b_k}\right|} \tag{4}$$

For k = n-1, (3) is the Newton iteration method, which requires $b_{n-1} \neq 0$. The above iteration method works in case of multiple roots.

Bisection step

The iteration method (3) does not guarantee monotonic convergence. If the condition

$$\left| P(z_{i+1}) \right| < \left| P(z_i) \right| \tag{5}$$

fails for some i, then a new approximation $\hat{z}_m$ is found such that

$$\left| P(\hat{z}_m) \right| < \left| P(z_i) \right| \tag{6}$$

The existence of a $\hat{z}_m$ satisfying (6) follows from $\left| P(z_i) \right| > 0$ and the maximum modulus principle. In fact, a suitable $\hat{z}_m$ can be found in the sequence

$$\hat{z}_m = z_i + 2^{-m} r_k \sqrt[(n-lm)]{\frac{-b_n}{b_{l_m}} \left| \frac{b_{l_m}}{b_n} \right|} \qquad m = 1, 2, \dots \tag{7}$$

where l_m is chosen so that

$$\left| b_{l_m} \right| (2^{-m} r_k)^{n-l_m} = \max \left[\left| b_j \right| (2^{-m} r_k)^{n-j} \right]$$

$$l_{m-1} \leq j \leq n-1 \qquad (8)$$

The proof of this is given in the first reference above.

Stopping criterion

The iteration method (3) is terminated if, at some step, the polynomial value does not decrease and the value itself is already less than an estimate of the roundoff error. If the estimated roundoff bound cannot be met by the polynomial value because of failure of the bisection method, the iteration is stopped with error indication ERROR='C'.

Estimate for roundoff error

The polynomial value

$$P(Z) = \sum_{r=0}^{n} a_r z^{n-r} \qquad (9)$$

is evaluated using nested multiplication:

$$b_{-1} = 0, \quad b_k = zb_{k-1} = a_k \quad \text{for} \quad k = 0, 1, 2, \ldots, n \qquad (10)$$

with $P(z) = b_n$.

Since all arithmetic operations are performed with floating point arithmetic, instead of the numbers b_k, internal approximations $\hat{b}_k$ will be generated that do not satisfy $P(z) = b_n$.

The following calculation will give an estimate of

$$\left| P(Z) - \hat{b}_n \right| .$$

The approximate values

$$\hat{b}_k = \underline{r\hat{b}}_k + i\, \underline{c\hat{b}}_k,$$

where $\underline{r\hat{b}}_k$ and $\underline{c\hat{b}}_k$ are the real and imaginary parts of $\hat{b}_k$, satisfy the equations.

$$\underline{r\hat{b}}_k = \left[\xi \cdot \underline{r\hat{b}}_{k-1} (1 + \pi_{1,k}) - \eta\, \underline{c\hat{b}}_{k-1} (1 + \pi_{2,k}) \right]$$

$$\left[(1 + \sigma_{1,k}) + \underline{ra}_k \right] / (1 + \sigma_{2,k})$$

$$\underline{c\hat{b}}_k = \left[\xi \circ \underline{c\hat{b}}_{k-1} (1 + \pi_{3,k}) + \eta \circ \underline{r\hat{b}}_{k-1} (1 + \pi_{4,k}) \right]$$

$$\left[(1 + \sigma_{3,k}) + \underline{ca}_k) \right] / (1 + \sigma_{4,k}) \qquad (11)$$

where $z = \xi + i\eta$, $Q_k = \underline{ra}_k + i\underline{ca}_k$, and $\sigma_{i,k}$, $\pi_{i,k}$ are relative errors of addition and multiplication respectively.

Solving (10) for a_k and inserting into

$$P(z) = \sum_{r=0}^{n} a_r z^{n-r}$$

gives

$$\begin{aligned} P(z) - \hat{b}_n = \sum_{k=0}^{n} z^{n-k} (\sigma_{2,k} \cdot \underline{r\hat{b}}_k &+ i\sigma_{4,k} \underline{c\hat{b}}_k \\ &- \xi\, \underline{r\hat{b}}_{k-1} (\pi_{1,k} + \sigma_{1,k} + \pi_{1,k}\, \sigma_{1,k}) \\ &+ \eta \underline{c\hat{b}}_{k-1} (\pi_{2,k} + \sigma_{1,k} + \pi_{2,k}\, \sigma_{1,k}) \\ &- i\, \xi\, \underline{c\hat{b}}_{k-1} (\pi_{3,k} + \sigma_{3,k} + \pi_{3,k}\, \sigma_{3,k}) \\ &- i\eta\, \underline{r\hat{b}}_{k-1} (\pi_{4,k} + \sigma_{3,k} + \pi_{4,k}\, \sigma_{3,k})) \end{aligned} \qquad (12)$$

With $\left| \sigma_{i,k} \right| \leq \sigma$, $\left| \pi_{i,k} \right| \leq \pi$, $\left| \pi_{i,k} (1 + \sigma_{i,k}) \right| \leq \pi$,

and $b_{-1} = 0$

$$\left| P(z) - \hat{b}_n \right| \leq \sum_{k=1}^{n-1} \left| z \right|^{n-k} \sigma \left| \hat{b}_k \right| + \left| z \right| \left| \hat{b}_{k-1} \right| (\sigma + 3\pi) + \sigma \left| b_0 \right| \left| z \right|^n \qquad (13)$$

or

$$\left| P(z) - \hat{b}_n \right| \leq \sum_{k=1}^{n-1} \left| z \right|^{n-k} \left| \hat{b}_k \right| (2\sigma + 3\pi) + \sigma \left(\left| b_0 \right| \left| z \right|^n + \left| b_n \right| \right) = E$$

E may be generated using the iteration scheme

$$e_0 = \frac{\sigma}{2\sigma + 3\pi}\left| b_0 \right| , \; e_k = \left| \hat{b}_k \right| + \left| z \right| e_{k-1} \text{ for}$$

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

giving

$$E = (2\sigma + 3\,\pi)\, e_n - (\sigma + 3\,\pi)\left| b_n \right| \tag{14}$$

In single precision, $\sigma = \pi = 10^{-6}$.

In double precision, $\sigma = \pi = 0.25 \cdot 10^{-15}$.

Programming Considerations:

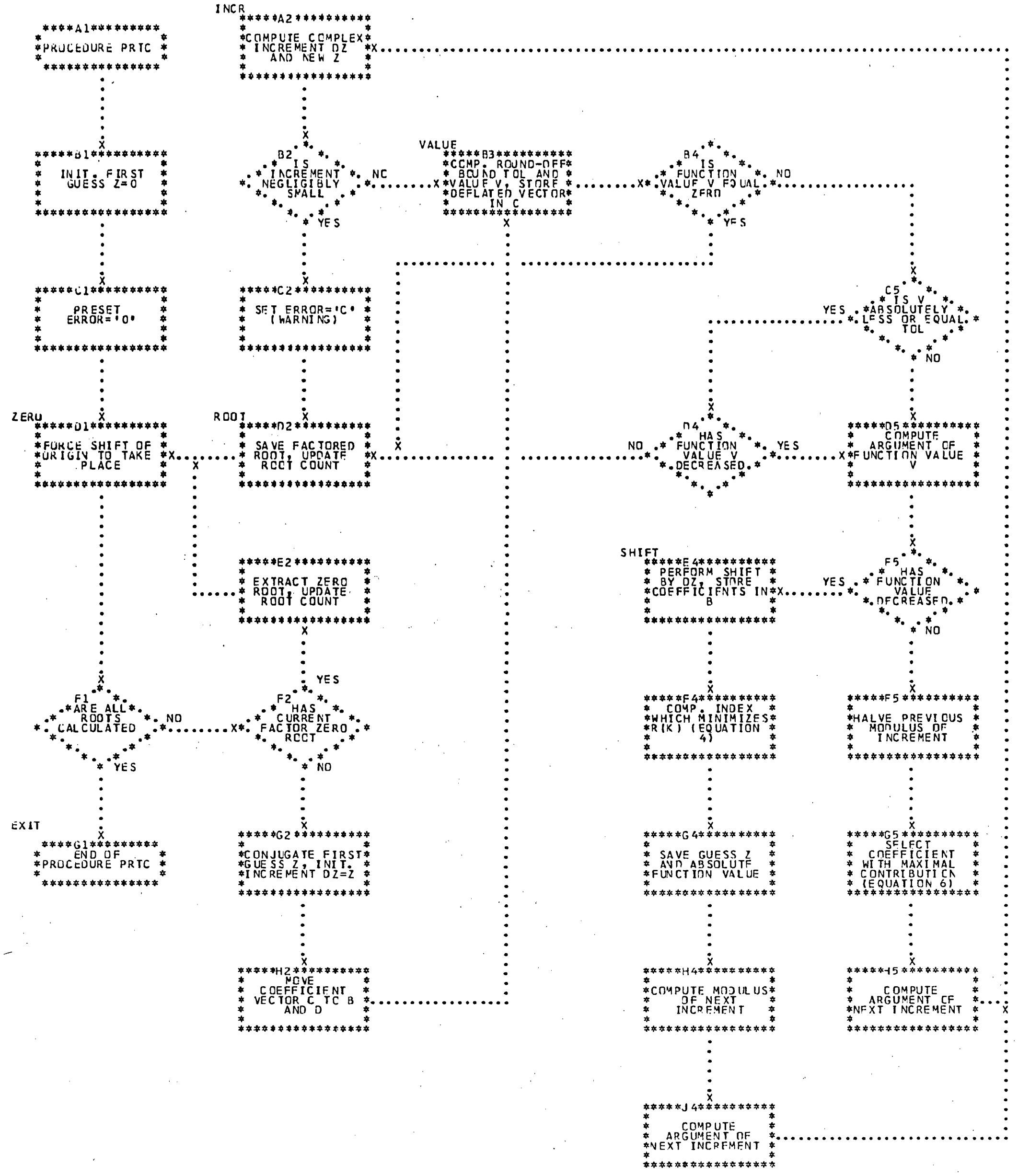
The polynomial must be given in normalized form; that is, the coefficient of z^n must be unity. Coefficients are ordered in decreasing order. Calculated zeros replace the coefficient vector; that is, the root stored in C(n) is calculated first and the root stored in C(1) is calculated last.

The iteration scheme starts with z = 0 initially.

As soon as the root z_1 has been calculated, P(z) is divided by $z - z_1$, giving $P_1(z)$. The complex conjugate $\overline{z}_1$ is used as the initial guess for a root of $P_1(z)$. Finally z_n is obtained as the root of $P_{n-1}(z)$, a linear polynomial.

No attempt is made to refine the approximated zeros with the original coefficient vector.

PROCEDURE PRTC CALCULATES ALL ROOTS OF A COMPLEX POLYNOMIAL


A1
PROCEDURE PRTC
B1
INIT. FIRST GUESS Z=0
C1
PRESET ERROR='0'
ZERO
D1
FORCE SHIFT OF ORIGIN TO TAKE PLACE
F1
ARE ALL ROOTS CALCULATED
NO
YES
EXIT
G1
END OF PROCEDURE PRTC
INCR
A2
COMPUTE COMPLEX INCREMENT DZ AND NEW Z
B2
IS INCREMENT NEGLIGIBLY SMALL
NC
YES
C2
SET ERROR='C' (WARNING)
ROOT
D2
SAVE FACTORED ROOT, UPDATE ROOT COUNT
E2
EXTRACT ZERO ROOT, UPDATE ROOT COUNT
YES
F2
HAS CURRENT FACTOR ZERO ROOT
NO
G2
CONJUGATE FIRST GUESS Z, INIT. INCREMENT DZ=Z
H2
MOVE COEFFICIENT VECTOR C TO B AND D
VALUE
B3
COMP. ROUND-OFF BOUND TOL AND VALUE V, STORE DEFLATED VECTOR IN C
B4
IS FUNCTION VALUE V EQUAL ZERO
NO
YES
D4
HAS FUNCTION VALUE V DECREASED
NO
YES
SHIFT
E4
PERFORM SHIFT BY DZ, STORE COEFFICIENTS IN B
F4
COMP. INDEX WHICH MINIMIZES R(K) (EQUATION 4)
G4
SAVE GUESS Z AND ABSOLUTE FUNCTION VALUE
H4
COMPUTE MODULUS OF NEXT INCREMENT
J4
COMPUTE ARGUMENT OF NEXT INCREMENT
C5
IS V ABSOLUTELY LESS OR EQUAL TOL
YES
NO
D5
COMPUTE ARGUMENT OF FUNCTION VALUE V
E5
HAS FUNCTION VALUE DECREASED
YES
NO
F5
HALVE PREVIOUS MODULUS OF INCREMENT
G5
SELECT COEFFICIENT WITH MAXIMAL CONTRIBUTION (EQUATION 6)
H5
COMPUTE ARGUMENT OF NEXT INCREMENT

Numerical Quadrature

Quadrature of Tabulated Functions

- Subroutine QTFG/QTFE

```
QTFG..
/**********************************************************************/
/*                                                                    */
/*       INTEGRATION CF A MONOTONICALLY TABULATED FUNCTION BY         */
/*       TRAPEZOIDAL RULE                                             */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,Y,Z,DIM),.
   DECLARE
         (X(*),Y(*),Z(*),SUM,XC,XN,YO,YN,H,HH)
         BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/
 /*      BINARY FLCAT(53),               /*DOUBLE PRECISION VERSION  /*D*/
         (DIM,I) BINARY FIXED,
         (ERROR EXTERNAL,SW)CHARACTER(1),.
   SW    ='1',.
   XO    =X(1),.
   GOTO COM,.
QTFE..
/**********************************************************************/
/*                                                                    */
/*       INTEGRATICN CF AN EQLIDISTANTLY TABULATED FUNCTION BY        */
/*       TRAPEZOIDAL RULE                                             */
/*                                                                    */
/**********************************************************************/
   ENTRY(H,Y,Z,DIM),.
   SW    ='0',.
   HH    =0.5*H,.
CCM..
   ERROR='1',.                           /*PRESET ERROR PARAMETER     */
   IF DIM GT 0                           /*NO ACTION IN CASE DIM LT 1 */
   THEN DO,.
         ERROR='0',.
         SUM   =0,.
         YO    =-Y(1),.
               DO I=1 TC DIM,.
               IF SW='1'
               THEN DO,.                 /*CALCULATE LENGTH OF INTERVAL */
                     XN    =X(I),.
                     HH    =C.5*(XN-XO),.
                     XO    =XN,.
                     END,.
               YN    =Y(I),.
               SUM   =SUM+HH*(YN+YO),.   /*ACCUMULATE INTEGRAL VALUE  */
               Z(I) =SUM,.
               YO    =YN,.
               END,.
         END,.
   END,.                                 /*END OF PROCEDURE QTFG      */
```

Purpose:

QTFG computes a vector Z of integral values for a given vector X of argument values and a given vector Y of function values.

Usage:

CALL QTFG (X, Y, Z, DIM);

X(DIM) - BINARY FLOAT [(53)]
Given vector of argument values.
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.
DIM - BINARY FIXED
Given dimension of vectors X, Y, Z.

Purpose:

QTFE computes a vector Z of integral values for a given vector X of equidistantly tabulated argument values and a given vector Y of function values.

Usage:

CALL QTFE (H, Y, Z, DIM);

H - BINARY FLOAT [(53)]
Given difference of two successive arguments:
$H = x_i - x_{i-1}$
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.
DIM - BINARY FIXED
Given dimension of vectors Y, Z.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='1' - means DIM is less than 1.
The vectors Z and Y may be identically allocated, which means that the given function values are replaced by the resultant integral values.

Method:

The integral values are obtained by means of the trapezoidal rule.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 75.

Mathematical Background:

Let x_i, y_i be the given table of arguments and function values.

The vector of integral values

$$z_i = \int_{x_1}^{x_i} y(x)\,dx$$

is calculated using the trapezoidal rule

$$z_i = z_{i-1} + \frac{(x_i - x_{i-1})}{2}(y_i + y_{i-1})$$

for i = 2,..., DIM

with $z_1 = 0$.

In case of equidistant arguments: $x_i - x_{i-1} = h$.

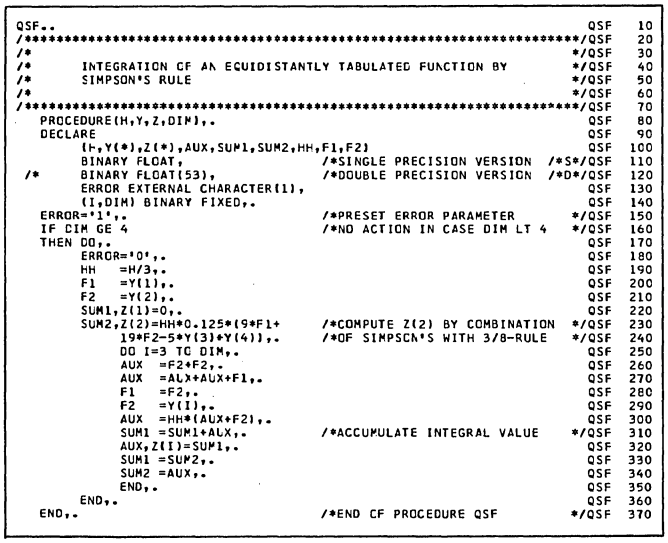
The local truncation error at each step is

$$R_i - \frac{1}{12}(x_i - x_{i-1})^3 \, y''(\xi_i), \left(\xi_i \in \left[x_i, x_{i-1}\right]\right)$$

assuming that y(x) has continuous derivatives up to the second order.

The total truncation error is the accumulation of the local errors at the previous step.

- Subroutine QSF



```
QSF..
/*********************************************************************/
/*                                                                   */
/*      INTEGRATION CF AN ECUIDISTANTLY TABULATEC FUNCTION BY       */
/*      SIMPSON'S RULE                                               */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(H,Y,Z,DIM),.
   DECLARE
        (H,Y(*),Z(*),AUX,SUM1,SUM2,HH,F1,F2)
        BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),               /*DOUBLE PRECISION VERSION  /*D*/
        ERROR EXTERNAL CHARACTER(1),
        (I,DIM) BINARY FIXED,.
   ERROR='1',.                          /*PRESET ERROR PARAMETER       */
   IF CIM GE 4                          /*NO ACTION IN CASE DIM LT 4   */
   THEN DO,.
        ERROR='0',.
        HH   =H/3,.
        F1   =Y(1),.
        F2   =Y(2),.
        SUM1,Z(1)=0,.
        SUM2,Z(2)=HH*0.125*(9*F1+       /*COMPUTE Z(2) BY COMBINATION  */
             19*F2-5*Y(3)+Y(4)),.       /*OF SIMPSCN'S WITH 3/8-RULE   */
             DO I=3 TC DIM,.
             AUX  =F2+F2,.
             AUX  =AUX+AUX+F1,.
             F1   =F2,.
             F2   =Y(I),.
             AUX  =HH*(AUX+F2),.
             SUM1 =SUM1+AUX,.           /*ACCUMULATE INTEGRAL VALUE    */
             AUX,Z(I)=SUM1,.
             SUM1 =SUM2,.
             SUM2 =AUX,.
             END,.
        END,.
   END,.                                /*END CF PROCEDURE QSF         */
```

Purpose:

QSF computes a vector Z of integral values, given a vector Y of function values corresponding to a vector X of equidistantly tabulated arguments.

Usage:

CALL QSF (H, Y, Z, DIM);

H - BINARY FLOAT [(53)]
Given difference of two successive arguments:
$H = x_i - x_{i-1}$

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.

DIM - BINARY FIXED
Given dimension of vectors Y and Z.

REMARKS:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='1' - means DIM is less than four.
Vectors Y and Z may be identically allocated, which means that the given function values are replaced by the resultant integral values.

Method:

The integral values z_i are obtained by Simpson's rule together with Newton's 3/8 rule.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 71-76.

R. Zurmühl, Praktische Mathematik für Ingenieure und Physiker. Springer, Berlin/Göttingen/Heidelberg, 1963, pp. 214-221.

Mathematical Background:

Let $Y = (y_1, y_2, \ldots, y_{DIM})$ be the given vector of function values corresponding to equidistant arguments x_i.

The vector of integral values

$$z_i = \int_{x_1}^{x_i} y(x)\ dx$$

is calculated from Simpson's rule

$$z_i = z_{i-2} + \frac{h}{3}(y_{i-2} + 4y_{i-1} + y_i) \text{ for } i = 3, \ldots, \text{ DIM} \qquad (1)$$

where the value of z_2 is obtained using a combination of Simpson's rule and Newton's 3/8 rule

$$z_i = z_{i-3} + 3/8\ h\ (y_{i-3} + 3y_{i-2} + 3y_{i-1} + y_i) \qquad (2)$$

resulting in

$$z_2 = z_1 + \frac{h}{24}(9y_1 + 19y_2 - 5y_3 + y_4) \qquad (3)$$

with $z_1 = 0$.

The local truncation errors of the above formulas are:

$$R_{1,i} = \frac{3}{90} h^5 y^{(4)}(\xi_i), \quad (\xi_i \in [x_{i-2}, x_i])$$

$$R_{2,i} = \frac{3}{80} h^5 y^{(4)}(\xi_i), \quad (\xi_i \in [x_{i-3}, x_i])$$

However, these truncation errors may accumulate.

- Subroutine QHFG/QHSG/QHFE/QHSE

```
QHFG..
/*******************************************************************/
/*                                                                 */
/*      INTEGRATION OF A MONOTONICALLY TABULATED FUNCTION WITH     */
/*      FIRST DERIVATIVE BY A HERMITIAN FORMULA OF FIRST ORDER     */
/*                                                                 */
/*******************************************************************/
   PROCEDURE(X,Y,FDY,Z,DIM),
   DECLARE
         (X(*),Y(*),Z(*),FDY(*),SDY(*),XC,XN,YC,YN,FDYO,FDYN,SDYO,SDYN,
         SUM1,SUM2,FACT,H,HH,HHH)
         BINARY FLOAT,              /*SINGLE PRECISION VERSION  /*S*/
 /*      BINARY FLOAT(53),          /*DOUBLE PRECISION VERSION  /*D*/
         (I,DIM) BINARY FIXED,
         (ERROR EXTERNAL,SW)CHARACTER(1),.
   SW   ='1',.
   GOTO MONO,.
QHSG..
/*******************************************************************/
/*                                                                 */
/*      INTEGRATION OF A MONOTONICALLY TABULATED FUNCTION WITH     */
/*      FIRST AND SECOND DERIVATIVES BY A HERMITIAN FORMULA OF     */
/*      SECOND ORDER                                               */
/*                                                                 */
/*******************************************************************/
   ENTRY(X,Y,FDY,SDY,Z,DIM),.
   SW   ='2',.
MONO..
   XO   =X(1),.
   GOTO MONEQ,.
QHFE..
/*******************************************************************/
/*                                                                 */
/*      INTEGRATION OF AN EQUIDISTANTLY TABULATED FUNCTION WITH    */
/*      FIRST DERIVATIVE BY A HERMITIAN FORMULA OF FIRST ORDER     */
/*                                                                 */
/*******************************************************************/
   ENTRY(H,Y,FDY,Z,DIM),.
   SW   ='3',.
   GOTO EQUI,.
QHSE..
/*******************************************************************/
/*                                                                 */
/*      INTEGRATION OF AN EQUIDISTANTLY TABULATED FUNCTION WITH    */
/*      FIRST AND SECOND DERIVATIVES BY A HERMITIAN FORMULA OF     */
/*      SECOND ORDER                                               */
/*                                                                 */
/*******************************************************************/
   ENTRY(H,Y,FDY,SDY,Z,DIM),.
   SW   ='4',.
EQUI..
   HH   =0.5*H,.
MONEQ..
   ERROR='1',.                      /*PRESET ERROR PARAMETER       */
   FACT =3.333333333333333E-01,.
   IF DIM GT 0                      /*NO ACTION IN CASE DIM LT 1   */
   THEN DO,.
        ERROR='0',.
        IF SW NE '1'
        THEN DO,.
             IF SW NE '3'
             THEN DO,.
                  FACT =0.4,.
                  SDYO =-SDY(1),.
                  END,.
             END,.
        YO   =-Y(1),.
        FDYO =FDY(1),.
        SUM1,SUM2=0,.
             DO I=1 TO DIM,.
             YN   =Y(I),.
             FDYN =FDY(I),.
             IF SW NE '3'
             THEN DO,.
                  IF SW NE '4'       /*SW ='1' OR SW ='2'          */
                  THEN DO,.          /*FOR NONEQUIDISTANT ARGUMENTS */
                       XN   =X(I),.  /*COMPUTE LENGTH OF INTERVAL   */
                       HH   =0.5*(XN-XO),.
                       XO   =XN,.
                       END,.
                  IF SW NE '1'       /*SW ='2' OR SW ='4'          */
                  THEN DO,.
                       SDYN =SDY(I),.
                       SUM2 =HH*HH*   /*MODIFY TO SECOND ORDER      */
                            (SDYO+    /*FORMULA                     */
                            SDYN)/15,.
                       SDYO =SDYN,.
                       END,.
                  END,.
             HHH  =HH*FACT,.
             SUM1 =SUM1+HH*(YO+YN+    /*ACCUMULATE INTEGRAL VALUE   */
                  HHH*(FDYO-FDYN)+SUM2),.
             Z(I) =SUM1,.
             YO   =YN,.
             FDYO =FDYN,.
             END,.
        END,.
   END,.                             /*END OF PROCEDURE QHFG       */
```

Purpose:

QHFG computes a vector Z of integral values for given vectors X, Y, and FDY of argument, function, and first derivative values respectively.

Usage:

CALL QHFG (X, Y, FDY, Z, DIM);

X(DIM) - BINARY FLOAT [(53)]
Given vector of argument values.
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
FDY(DIM) - BINARY FLOAT [(53)]
Given vector of first derivative values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.
DIM - Given dimension of vectors X, Y, FDY, Z.

Purpose:

QHSG computes a vector Z of integral values for given vectors X, Y, FDY, and SDY of argument, function, first derivative, and second derivative values respectively.

Usage:

CALL QHSG (X, Y, FDY, SDY, Z, DIM);

X(DIM) - BINARY FLOAT [(53)]
Given vector of arguments.
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
FDY(DIM) - BINARY FLOAT [(53)]
Given vector of first derivative values.
SDY(DIM) - BINARY FLOAT [(53)]
Given vector of second derivative values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.
DIM - BINARY FIXED
Given dimension of vectors X, Y, FDY, SDY, Z.

Purpose:

QHFE computes a vector Z of integral values for given vectors Y and FDY of function and first derivative values respectively, corresponding to a vector X of equidistantly tabulated argument values.

Usage:

CALL QHFE (H, Y, FDY, Z, DIM);

H - BINARY FLOAT [(53)]
Given difference of two arguments:
$H = x_i - x_{i-1}$
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
FDY(DIM) - BINARY FLOAT [(53)]
Given vector of first derivative values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.
DIM - BINARY FIXED
Given dimensions of vectors Y, FDY, Z.

Purpose:

QHSE computes a vector Z of integral values for given vectors Y, FDY, SDY of function values, first derivative values, and second derivative values respectively, corresponding to a vector X of equidistantly tabulated arguments.

Usage:

CALL QHSE (H, Y, FDY, SDY, Z, DIM);

H - BINARY FLOAT [(53)]
Given difference of two argument values: $H = x_i - x_{i-1}$
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
FDY(DIM) - BINARY FLOAT [(53)]
Given vector of first derivative values.
SDY(DIM) - BINARY FLOAT [(53)]
Given vector of second derivative values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of integral values.
DIM - BINARY FIXED
Given dimensions of vectors Y, FDY, SDY, Z.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:
ERROR ='1' means DIM is less than 1.

The storage allocation of vector Z may be identical to one of the given vectors, which means that the given values are replaced by the resultant integral values.

Method:

The calculation of integral values is done using Hermitian formulas of the first and second order.

For reference see:

F.B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 314-319.

R. Zurmühl, Praktische Mathematik für Ingenieure und Physiker. Springer, Berlin/Göttingen/Heidelberg, 1963, pp. 227-230.

Mathematical Background:

Let X, Y, FDY, SDY denote the vectors of arguments x_i, function values y_i, first derivative values y_i' and second derivative values y_i'' respectively.

The vector of integral values

$$z_i = \int_{x_1}^{x_i} y(x)\,dx$$

is calculated from one of the following:

Hermitian formula of first order:

$$z_i = z_{i-1} + \frac{x_i - x_{i-1}}{2} \Big[y_{i-1} + y_i + \frac{x_i - x_{i-1}}{6} (y'_{i-1} - y'_i) \Big] \qquad (1)$$

with $z_1 = 0$. $(i = 2, 3, \ldots, \text{DIM})$

Hermitian formula of second order:

$$z_i = z_{i-1} + \frac{x_i - x_{i-1}}{2} \Big\{ y_{i-1} + y_i + \frac{x_i - x_{i-1}}{5} \Big[y_{i-1}' - y_i' + \frac{x_i - x_{i-1}}{12} (y_{i-1}'' + y_i'') \Big] \Big\} \qquad (2)$$

$(i = 2, 3, \ldots, \text{DIM})$

with $z_1 = 0$.

Corresponding formulas for equidistant arguments (meaning $x_i - x_{i-1} = h$):

$$z_i = z_{i-1} + \frac{h}{2} \Big[y_{i-1} + y_i + \frac{h}{6} (y'_{i-1} - y_i') \Big] \qquad (1a)$$

$(i = 2, 3, \ldots, \text{DIM})$

with $z_1 = 0$, and

$$z_i = z_{i-1} + \frac{h}{2} \Big\{ y_{i-1} + y_i + \frac{h}{5} \Big[y'_{i-1} - y'_i + \frac{h}{12} (y''_{i-1} + y_i'') \Big] \Big\} \qquad (2a)$$

$(i = 2, 3, \ldots, \text{DIM})$

with $z_1 = 0$.

Assuming that $y(x)$ has continuous derivatives up to the sixth order, the local truncation error at each step is

$$R_{1,i} = \frac{(x_i - x_{i-1})^5}{120} y^{(4)} (\xi_i)$$

$$(\xi_i \in [x_{i-1}, x_i])$$

and

$$R_{2,i} = \frac{(x_i - x_{i-1})^7}{100800} y^{(6)} (\xi_i),$$

$$(\xi_i \in [x_{i-1}, x_i])$$

The total truncation error is the accumulation of the local errors at the previous step.

For equidistant arguments, this leads to:

$$R_{1n} = \frac{1}{120} h^4 y^{(4)} (\xi),$$

$$(\xi \in [x_1, x_n])$$

and

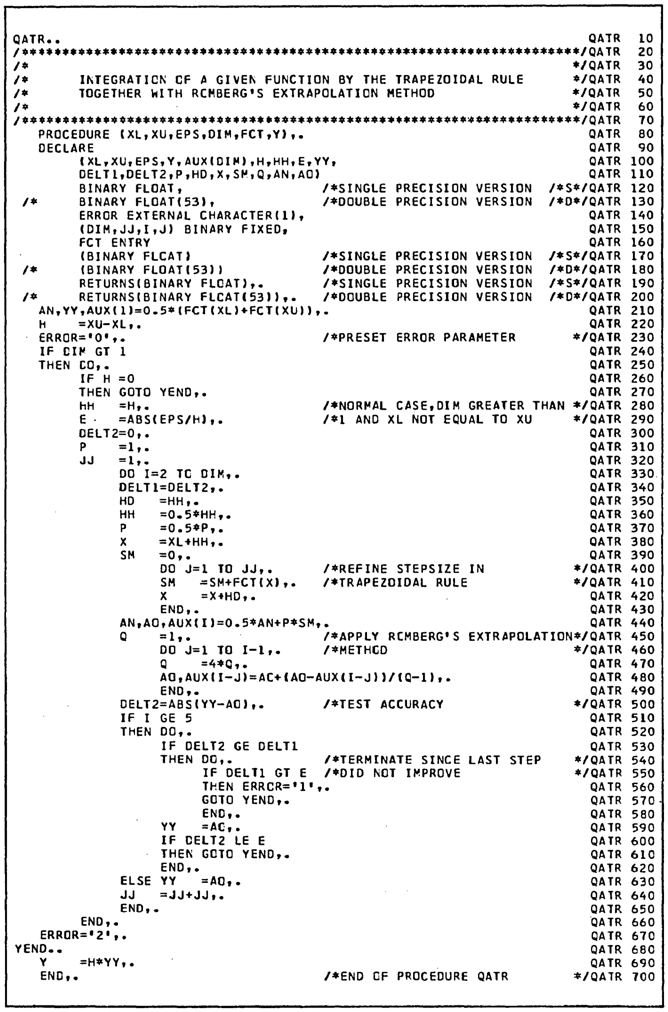
$$R_{2n} = -\frac{1}{100800} h^6 y^{(6)} (\xi),$$

$$(\xi \in [x_1, x_n])$$

where 1 is the length of the integration interval.

Quadrature of Nontabulated Functions

- Subroutine QATR



```
QATR..
/***********************************************************************************/
/*                                                                                 */
/*        INTEGRATICN CF A GIVEN FUNCTION BY THE TRAPEZOIDAL RULE                  */
/*        TOGETHER WITH RCMBERG'S EXTRAPOLATION METHOD                             */
/*                                                                                 */
/***********************************************************************************/
    PROCEDURE (XL,XU,EPS,DIM,FCT,Y),.
    DECLARE
          (XL,XU,EPS,Y,AUX(DIM),H,HH,E,YY,
          DELT1,DELT2,P,HD,X,SM,Q,AN,AO)
          BINARY FLOAT,                     /*SINGLE PRECISION VERSION      /*S*/
 /*       BINARY FLOAT(53),                 /*DOUBLE PRECISION VERSION      /*D*/
          ERROR EXTERNAL CHARACTER(1),
          (DIM,JJ,I,J) BINARY FIXED,
          FCT ENTRY
          (BINARY FLCAT)                    /*SINGLE PRECISION VERSION      /*S*/
 /*       (BINARY FLOAT(53))                /*DOUBLE PRECISION VERSION      /*D*/
          RETURNS(BINARY FLOAT),.           /*SINGLE PRECISION VERSION      /*S*/
 /*       RETURNS(BINARY FLCAT(53)),.       /*DOUBLE PRECISION VERSION      /*D*/
    AN,YY,AUX(1)=0.5*(FCT(XL)+FCT(XU)),.
    H    =XU-XL,.
    ERROR='0',.                             /*PRESET ERROR PARAMETER             */
    IF CIM GT 1
    THEN CO,.
          IF H =0
          THEN GOTO YEND,.
          HH   =H,.                         /*NORMAL CASE,DIM GREATER THAN       */
          E    =ABS(EPS/H),.                /*1 AND XL NOT EQUAL TO XU           */
          DELT2=0,.
          P    =1,.
          JJ   =1,.
                DO I=2 TC DIM,.
                DELT1=DELT2,.
                HD    =HH,.
                HH    =0.5*HH,.
                P     =0.5*P,.
                X     =XL+HH,.
                SM    =0,.
                      DO J=1 TO JJ,.        /*REFINE STEPSIZE IN                 */
                      SM    =SM+FCT(X),.    /*TRAPEZOIDAL RULE                   */
                      X     =X+HD,.
                      END,.
                AN,AO,AUX(I)=0.5*AN+P*SM,.
                Q     =1,.                  /*APPLY RCMBERG'S EXTRAPOLATION*/
                      DO J=1 TO I-1,.       /*METHCD                             */
                      Q     =4*Q,.
                      AO,AUX(I-J)=AC+(AO-AUX(I-J))/(Q-1),.
                      END,.
                DELT2=ABS(YY-AO),.          /*TEST ACCURACY                      */
                IF I GE 5
                THEN DO,.
                      IF DELT2 GE DELT1
                      THEN DO,.             /*TERMINATE SINCE LAST STEP          */
                            IF DELT1 GT E   /*DID NOT IMPROVE                    */
                            THEN ERRCR='1',.
                            GOTO YEND,.
                            END,.
                      YY    =AC,.
                      IF CELT2 LE E
                      THEN GCTO YEND,.
                      END,.
                ELSE YY    =AO,.
                JJ    =JJ+JJ,.
                END,.
          END,.
    ERROR='2',.
YEND..
    Y    =H*YY,.
    END,.                                   /*END CF PROCEDURE QATR              */
```

Purpose:

QATR computes the integral value

$$Y = \int_{XL}^{XU} FCT(X)\, dX$$

for a given function FCT(X), defined in the closed interval [XL, XU], by the trapezoidal rule together with Romberg's extrapolation method.

Usage:

CALL QATR (XL, XU, EPS, DIM, FCT, Y);

XL -	BINARY FLOAT [(53)] Given lower bound of the interval.
XU -	BINARY FLOAT [(53)] Given upper bound of the interval.
EPS -	BINARY FLOAT [(53)] Given upper bound of the absolute error.
DIM -	BINARY FIXED Given maximum number of extrapolation steps +1 (for details see "Programming Considerations").
FCT -	ENTRY Given procedure for calculation of the function values, which must be supplied by the user.

Usage:
FCT(T)

T -	BINARY FLOAT [(53)] Given argument.
FCT(T) -	BINARY FLOAT [(53)] Resultant function value.

Y -	BINARY FLOAT [(53)] Resultant approximation for the integral value.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR = '1' means that it is impossible to reach the required accuracy because of rounding errors.

ERROR = '2' means that it was impossible to check accuracy because DIM is less than 5, or the required accuracy could not be reached within DIM-1 steps.

Method:

Evaluation of the approximation Y to the integral value is done by means of the trapezoidal rule combined with Romberg's extrapolation method.

For reference see:

S. 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.

Bauer, Algorithm 60, CACM, vol. 4, 155.6 (1961), pp. 255.

Mathematical Background:

The problem is to compute an approximation for

$$y = \int_a^b f(x)\,dx \tag{1}$$

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

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

$$f_{i,k} = f(x_{i,k}) \quad (k = 0,1,2, \ldots, 2^i)$$

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

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

Then the following can be written:

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

with unknown coefficients $C_{0,2r}$ that 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 $T_{0,i+1}$, we can generate an extrapolated value:

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

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 (equation 3), and:

$$T_{2,i} = T_{1,i+1} + \frac{T_{1,i+1} - T_{1,i}}{2^4 - 1} \tag{4}$$

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.

Programming Considerations:

The subroutine uses the scheme shown in Figure 1 for computation of T values and generates the upward diagonal in the one-dimensional storage array AUX, using the general formula:

$$T_{k,j} = T_{k-1,j+1} + \frac{T_{k-1,j+1} - T_{k-1,j}}{2^{2k} - 1} \tag{5}$$

($k+j = i$, $j = i-1, i-2, \ldots, 2,1,0$)

and storing:

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

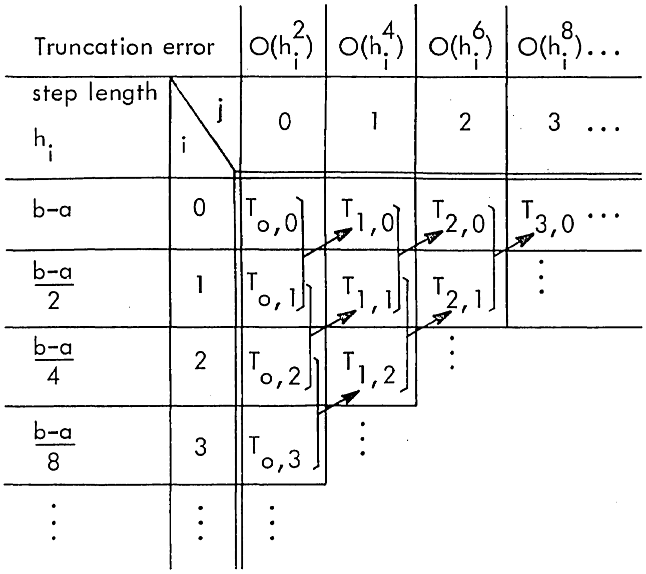
$T_{1,i-1}$ into AUX (i)

.

.

.

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



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

Figure 1. 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.

- <u>Subroutine</u> QGn (n = 2, 4, 8, 16, 24, 32, 48)

```
QG2..                                                                   QG2
/**********************************************************************/QG2
/*                                                                    */QG2
/*      INTEGRATICN CF GIVEN FUNCTICN BY 2-POINT GAUSSIAN             */QG2
/*      CUADRATURE FORMULA                                            */QG2
/*                                                                    */QG2
/**********************************************************************/QG2
   PROCEDURE(XL,XU,FCT,Y),.                                             QG2
   CECLARE                                                              QG2
        (XL,XU,Y,A,B)                                                   QG2
        BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/QG2
 /*     BINARY FLOAT (53),              /*DOUBLE PRECISION VERSION  /*D*/QG2
        FCT ENTRY RETURNS                                               QG2
        (BINARY FLOAT),.                /*SINGLE PRECISION VERSION  /*S*/QG2
 /*     (BINARY FLOAT (53)),.           /*DOUBLE PRECISION VERSION  /*D*/QG2
   A    =0.5*(XU+XL),.                                                  QG2
   B    =XU-XL,.                                                        QG2
   Y    =2.886751345948128E-01*B,.                                      QG2
   Y    =0.5*B*(FCT(A+Y)+FCT(A-Y)),.                                    QG2
   END,.                                /*END OF PROCEDURE QG2        */QG2
```

```
QG4..                                                                   QG4
/**********************************************************************/QG4
/*                                                                    */QG4
/*      INTEGRATICN OF A GIVEN FUNCTION BY 4-POINT GAUSSIAN           */QG4
/*      CUADRATURE FORMULA                                            */QG4
/*                                                                    */QG4
/**********************************************************************/QG4
   PROCEDURE (XL,XU,FCT,Y),.                                            QG4
   CECLARE                                                              QG4
        (XL,XU,Y,A,B,C)                                                 QG4
        BINARY FLCAT,                   /*SINGLE PRECISION VERSION  /*S*/QG4
 /*     BINARY FLCAT (53),              /*DOUBLE PRECISION VERSION  /*D*/QG4
        FCT ENTRY RETURNS                                               QG4
        (BINARY FLOAT),.                /*SINGLE PRECISION VERSION  /*S*/QG4
 /*     (BINARY FLOAT (53)),.           /*DOUBLE PRECISION VERSION  /*D*/QG4
   A    =0.5*(XU+XL),.                                                  QG4
   B    =XU-XL,.                                                        QG4
   C    =4.3C5681557S7C263E-01*B,.                                      QG4
   Y    =1.739274225687269E-01*(FCT(A+C)+FCT(A-C)),.                    QG4
   C    =1.699905217924281E-01*B,.                                      QG4
   Y    =B*(Y+3.260725774312731E-01*(FCT(A+C)+FCT(A-C))),.              QG4
   END,.                                /*END CF PROCEDURE QG4        */QG4
```

```
QG8..                                                                   QG8
/**********************************************************************/QG8
/*                                                                    */QG8
/*      INTEGRATICN OF A GIVEN FUNCTION BY 8-PCINT GAUSSIAN           */QG8
/*      CUADRATURE FCRMULA                                            */QG8
/*                                                                    */QG8
/**********************************************************************/QG8
   PRCCECURE(XL,XU,FCT,Y),.                                             QG8
   CECLARE                                                              QG8
        (XL,XU,Y,A,B,C)                                                 QG8
        BINARY FLCAT,                   /*SINGLE PRECISION VERSION  /*S*/QG8
 /*     BINARY FLCAT (53),              /*DOUBLE PRECISION VERSION  /*D*/QG8
        FCT ENTRY RETURNS                                               QG8
        (BINARY FLOAT),                 /*SINGLE PRECISION VERSION  /*S*/QG8
 /*     (BINARY FLOAT (53)),            /*DOUBLE PRECISION VERSION  /*D*/QG8
        LY BINARY FLOAT (53),                                           QG8
        X( 8) BINARY FLOAT (53) STATIC INITIAL                          QG8
        (4.8C1449282487681E-C1,     5.061426814518813E-02,              QG8
        3.983332387068134E-C1,     1.111905172266872E-01,               QG8
        2.627662C49581645E-C1,     1.568533229389436E-01,               QG8
        9.171732124782490E-02,     1.813418916891810E-01),.             QG8
   A    =0.5*(XU+XL),.                                                  QG8
   B    =XU-XL,.                                                        QG8
   LY   =0,.                                                            QG8
        CO I=1 TO  7 BY 2,.                                             QG8
        C    =X(I)*B,.                                                  QG8
        LY   =LY+X(I+1)*(FCT(A+C)+FCT(A-C)),.                           QG8
        END,.                                                           QG8
   Y    =LY*B,.                                                         QG8
   END,.                                /*END CF PROCEDURE QG8        */QG8
```

```
QG16..                                                                  QG16
/**********************************************************************/QG16
/*                                                                    */QG16
/*      INTEGRATICN CF A GIVEN FUNCTICN BY 16-PCINT GAUSSIAN          */QG16
/*      CUADRATURE FORMULA                                            */QG16
/*                                                                    */QG16
/**********************************************************************/QG16
   PROCEDURE(XL,XU,FCT,Y),.                                             QG16
   CECLARE                                                              QG16
        (XL,XU,Y,A,B,C)                                                 QG16
        BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/QG16
 /*     BINARY FLOAT (53),              /*DOUBLE PRECISION VERSION  /*D*/QG16
        FCT ENTRY RETURNS                                               QG16
        (BINARY FLOAT),                 /*SINGLE PRECISION VERSION  /*S*/QG16
 /*     (BINARY FLCAT (53)),            /*DOUBLE PRECISION VERSION  /*D*/QG16
        LY BINARY FLCAT (53),                                           QG16
        X(16) BINARY FLOAT (53) STATIC INITIAL                          QG16
        (4.947004674958250E-01,     1.357622970587705E-02,              QG16
        4.722875115366163E-01,     3.112676196932395E-02,               QG16
        4.328156011939159E-01,     4.757925584124639E-02,               QG16
        3.777022041775015E-C1,     6.231448562776694E-02,               QG16
        3.089381222013219E-01,     7.479799440828837E-02,               QG16
        2.290C83888286137E-C1,     8.457825969750127E-02,               QG16
        1.408017753896295E-C1,     9.130170752246179E-02,               QG16
        4.750625491881872E-C2,     9.472530522753425E-02),.             QG16
   A    =0.5*(XU+XL),.                                                  QG16
   B    =XU-XL,.                                                        QG16
   LY   =0,.                                                            QG16
        DO I=1 TO 15 BY 2,.                                             QG16
        C    =X(I)*B,.                                                  QG16
        LY   =LY+X(I+1)*(FCT(A+C)+FCT(A-C)),.                           QG16
        END,.                                                           QG16
   Y    =LY*B,.                                                         QG16
   END,.                                /*END CF PROCEDURE QG16       */QG16
```

```
QG24..                                                                  QG24  10
/**********************************************************************/QG24  20
/*                                                                    */QG24  30
/*      INTEGRATICN CF A GIVEN FUNCTICN BY 24-PCINT GAUSSIAN          */QG24  40
/*      CUADRATURE FCRMULA                                            */QG24  50
/*                                                                    */QG24  60
/**********************************************************************/QG24  70
   PRCCEDURE(XL,XU,FCT,Y),.                                             QG24  80
   CECLARE                                                              CG24  90
         (XL,XU,Y,A,B,C)                                                QG24 1C0
         BINARY FLCAT,                    /*SINGLE PRECISION VERSION  /*S*/QG24 110
 /*      BINARY FLCAT (53),               /*DOUBLE PRECISION VERSION  /*D*/QG24 120
         FCT ENTRY RETURNS                                              QG24 130
         (BINARY FLOAT),                  /*SINGLE PRECISION VERSION  /*S*/QG24 140
 /*      (BINARY FLCAT (53)),             /*DOUBLE PRECISION VERSION  /*D*/QG24 150
         LY BINARY FLCAT (53),                                          QG24 160
         X(24) BINARY FLCAT (53) STATIC INITIAL                         QG24 170
         (4.975936C99985107E-01,    6.170614899993600E-03,              QG24 180
         4.873642779856547E-C1,    1.426569431446683E-02,               QG24 190
         4.691372760013664E-C1,    2.213871940870990E-02,               QG24 200
         4.432077635C220C5E-C1,    2.964929245771839E-02,               QG24 210
         4.100009929869515E-01,    3.667324070554C15E-02,               QG24 220
         3.7C0620957892772E-C1,    4.309508076597664E-02,               QG24 230
         3.240468259684878E-C1,    4.880932605205694E-02,               QG24 240
         2.727107356944198E-C1,    5.372213505798282E-02,               QG24 250
         2.168967538130226E-C1,    5.775283402686280E-02,               QG24 260
         1.57521339848C817E-01,    6.083523646390170E-02,               QG24 270
         9.555943373680815E-02,    6.291872817341415E-02,               QG24 280
         3.202844643130281E-C2,    6.396909767337608E-02),.             QG24 290
   A     =0.5*(XU+XL),.                                                 QG24 300
   B     =XU-XL,.                                                       QG24 310
   LY    =0,.                                                           QG24 320
         DO I=1 TO 23 BY 2,.                                            QG24 330
         C     =X(I)*B,.                                                QG24 340
         LY    =LY+X(I+1)*(FCT(A+C)+FCT(A-C)),.                         QG24 350
         END,.                                                          QG24 360
   Y     =LY*B,.                                                        QG24 37C
   END,.                          /*END CF PROCEDURE QG24             */QG24 380
```

```
QG32..                                                                  QG32  10
/**********************************************************************/QG32  20
/*                                                                    */QG32  30
/*      INTEGRATION CF A GIVEN FUNCTION BY 32-PCINT GAUSSIAN          */QG32  40
/*      CUADRATURE FORMULA                                            */QG32  50
/*                                                                    */QG32  60
/**********************************************************************/QG32  70
   PROCEDURE(XL,XU,FCT,Y),.                                             QG32  80
   DECLARE                                                              QG32  90
         (XL,XU,Y,A,B,C)                                                QG32 100
         BINARY FLOAT,                    /*SINGLE PRECISION VERSION  /*S*/QG32 110
 /*      BINARY FLOAT (53),               /*DOUBLE PRECISION VERSION  /*D*/QG32 120
         FCT ENTRY RETURNS                                              QG32 130
         (BINARY FLOAT),                  /*SINGLE PRECISION VERSION  /*S*/QG32 140
 /*      (BINARY FLOAT (53)),             /*DOUBLE PRECISION VERSION  /*D*/QG32 150
         LY BINARY FLOAT (53),                                          QG32 160
         X(32) BINARY FLOAT (53) STATIC INITIAL                         QG32 170
         (4.9863193092474C8E-01,    3.509305004735048E-03,              QG32 180
         4.928057557726342E-C1,    8.137197365452835E-03,               QG32 190
         4.823811277937532E-01,    1.269603265463103E-02,               QG32 200
         4.674530379688698E-01,    1.713693145651072E-02,               QG32 210
         4.48160577883C261E-01,    2.141794901111334E-02,               QG32 220
         4.246838068662850E-C1,    2.549902963118809E-02,               QG32 230
         3.972418979839712E-01,    2.934204673926777E-02,               QG32 240
         3.66C910593701448E-C1,    3.291111138818092E-02,               QG32 250
         3.315221334651C76E-01,    3.617289705442425E-02,               QG32 260
         2.938578786203812E-C1,    3.909694789353515E-02,               QG32 270
         2.534499544661147E-01,    4.165596211347338E-02,               QG32 280
         2.106756380653177E-C1,    4.382604650220191E-02,               QG32 290
         1.659343011410638E-01,    4.558693934788194E-02,               QG32 300
         1.196436811260685E-C1,    4.692219954040228E-02,               QG32 310
         7.223598079139825E-C2,    4.781936003963743E-02,               QG32 320
         2.415383284386916E-C2,    4.827004425736390E-02),.             QG32 330
   A     =C.5*(XU+XL),.                                                 QG32 340
   B     =XU-XL,.                                                       QG32 350
   LY    =0,.                                                           QG32 360
         DO I=1 TO 31 BY 2,.                                            QG32 370
         C     =X(I)*B,.                                                QG32 380
         LY    =LY+X(I+1)*(FCT(A+C)+FCT(A-C)),.                         QG32 390
         END,.                                                          QG32 400
   Y     =LY*B,.                                                        QG32 410
   END,.                          /*END CF PROCEDURE QG32             */QG32 420
```

```
CG48..                                                                  QG48  10
/**********************************************************************/QG48  20
/*                                                                    */QG48  30
/*      INTEGRATION CF A GIVEN FUNCTICN BY 48-PCINT GAUSSIAN          */QG48  40
/*      CUADRATURE FCRMULA                                            */QG48  50
/*                                                                    */QG48  6C
/**********************************************************************/QG48  70
   PRCCEDURE(XL,XU,FCT,Y),.                                             QG48  80
   CECLARE                                                              QG48  90
         (XL,XU,Y,A,B,C)                                                QG48 100
         BINARY FLCAT,                    /*SINGLE PRECISION VERSION  /*S*/QG48 110
 /*      BINARY FLCAT (53),               /*DOUBLE PRECISION VERSION  /*D*/QG48 120
         FCT ENTRY RETURNS                                              QG48 130
         (BINARY FLOAT),                  /*SINGLE PRECISION VERSION  /*S*/QG48 140
 /*      (BINARY FLOAT (53)),             /*DOUBLE PRECISION VERSION  /*D*/QG48 150
         LY BINARY FLCAT (53),.                                         QG48 160
   CECLARE                                                              QG48 17C
         X(24) BINARY FLCAT(53) STATIC INITIAL (                        QG48 180
         4.993855036262131E-C1,   4.967650861331754E-01,                QG48 190
         4.920622918614134E-01,   4.852957962731236E-01,                QG48 200
         4.764938515802154E-C1,   4.656933453532772E-01,                QG48 210
         4.529395683577848E-C1,   4.382860101371239E-01,                QG48 220
         4.217941308121968E-01,   4.035331020147213E-01,                QG48 230
         3.835795162578702E-C1,   3.620170654619073E-01,                QG48 240
         3.389361898163320E-C1,   3.144336983882568E-01,                QG48 250
         2.886123630419864E-01,   2.615804873611165E-01,                QG48 260
         2.334514523754792E-01,   2.043432409953584E-01,                QG48 27C
         1.743779431460804E-C1,   1.436812436777278E-01,                QG48 280
         1.123818951973445E-01,   8.061117803444586E-02,                QG48 290
         4.850234960473135E-C2,   1.619008548143468E-02),.              QG48 300
   CECLARE                                                              QG48 310
         W(24) BINARY FLCAT(53) STATIC INITIAL (                        QG48 320
         1.576673026152919E-03,   3.663776950638131E-03,                QG48 330
         5.738617289617270E-03,   7.789657861471924E-03,                QG48 340
         9.808080228677764E-C3,   1.178538041966219E-02,                QG48 350
         1.371325485417847E-C2,   1.558361391639904E-02,                QG48 360
         1.738861128238522E-C2,   1.912067553291535E-02,                QG48 370
```

```
         2.077254147173237E-C2,   2.233728042834714E-02,                QG48 380
         2.380832924624524E-C2,   2.517951777692724E-02,                QG48 390
         2.644509474259683E-C2,   2.759975184999208E-02,                QG48 4C0
         2.8638646C502C161E-02,   2.955741984919782E-02,                QG48 410
         3.035221958294694E-C2,   3.1C1971157994633E-02,                QG48 420
         3.155709614312701E-C2,   3.196211929232409E-02,                QG48 430
         3.223308221797504E-02,   3.236884840634196E-02),.              QG48 440
   A     =0.5*(XU+XL),.                                                 QG48 45C
   B     =XU-XL,.                                                       QG48 46C
   LY    =C,.                                                           QG48 47C
         CO I=1 TO 24,.                                                 QG48 480
         C     =X(I)*B,.                                                QG48 490
         LY    =LY+W(I)*(FCT(A+C)+FCT(A-C)),.                           QG48 500
         END,.                                                          QG48 510
   Y     =LY*B,.                                                        QG48 520
   END,.                          /*END CF PROCEDURE QG48             */QG48 530
```

Purpose:

QGn computes the integral value Y $\int_{XL}^{XU}$ FCT(X) dX for a given function FCT (X) defined in the closed interval [XL, XU], using Gaussian quadrature formulas.

Usage:

CALL QGn (XL, XU, FCT, Y);

XL - BINARY FLOAT [(53)]
Given lower bound of the integral.

XU - BINARY FLOAT [(53)]
Given upper bound of the integral.

FCT - ENTRY
Given procedure for the computation of the function values, which must be supplied by the user.

Usage:

FCT(X)

FCT(X) - BINARY FLOAT [(53)]
Resultant function value.

X - BINARY FLOAT [(53)]
Given argument value.

Y - BINARY FLOAT [(53)]
Resultant integral value.

Remarks:

The number n within the procedure name QGn indicates the number of nodes used for calculation of Y.

Method:

Gaussian quadrature formulas are used for the evaluation of the integral values.

For reference see:

V. I. Krylow, Approximate Calculation of Integrals, Macmillan, New York-London, 1962, pp. 100-111 and 337-340.

Mathematical Background:

Set:

x_l = lower bound of integral
x_u = upper bound of integral
n = number of nodes used for the evaluation of the integral value.

By means of the linear transformation

$$x = t_0 + t_1 t$$

$$\text{with } t_0 = \frac{x_u + x_l}{2} \quad \text{and} \quad t_1 = \frac{x_u - x_l}{2} \tag{1}$$

the argument range $x_l \le x \le x_u$ is mapped onto

$$-1 \le t \le +1$$

and the integral

$$y = \int_{x_l}^{x_u} f(x)\,dx \tag{2}$$

is reduced to standard form

$$y = \int_{-1}^{+1} \varphi(t)\,dt \tag{3}$$

with $\varphi(t) = t_1 f(t_0 + t_1 t)$.

Gaussian quadrature formulas are used to compute (3).

The integral value y is approximated by a weighted sum of function values:

$$y^{(n)} = 2t_1 \sum_{k=1}^{n} \left\{ \frac{A_k^{(n)}}{2} f(t_0 + t_1 t_k^{(n)}) \right\}$$

The value $y^{(n)}$ is exact whenever f(x) is a polynomial of degree less than or equal to 2n-1.

The weights $A_k^{(n)}$ and nodes $t_k^{(n)}$ are symmetric with respect to the origin t = 0:

$$A_k^{(n)} = A_{n-k+1}^{(n)}, \quad t_k^{(n)} = -t_{n-k+1}^{(n)}$$

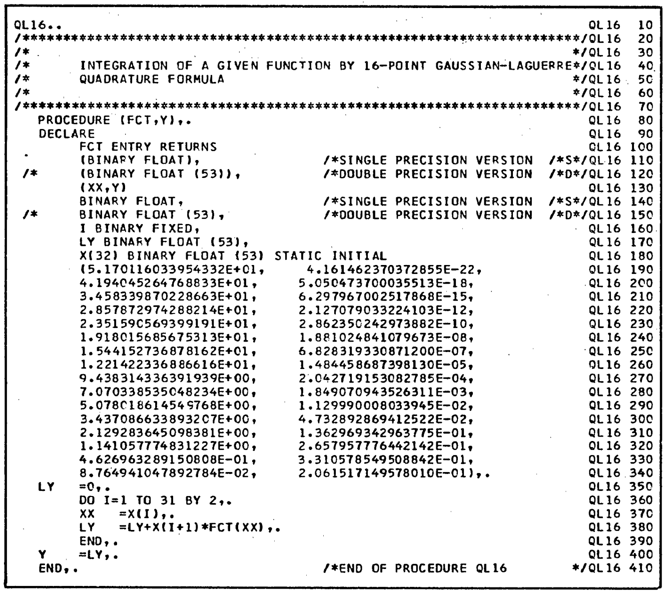
- Subroutine QLn (n = 2, 4, 8, 12, 16, 24)

```
QL2..
/**********************************************************************/
/*                                                                    */
/*      INTEGRATION OF A GIVEN FUNCTION BY 2-POINT GAUSSIAN-LAGUERRE */
/*      QUADRATURE FORMULA                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (X,Y)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
   X    =3.414213562373095E+00,.
   Y    =1.464466C94067262E-01*FCT(X),.
   X    =5.857864376269050E-01,.
   Y    =Y+8.535533905932738E-01*FCT(X),.
   END,.                               /*END OF PROCEDURE QL2        */
```

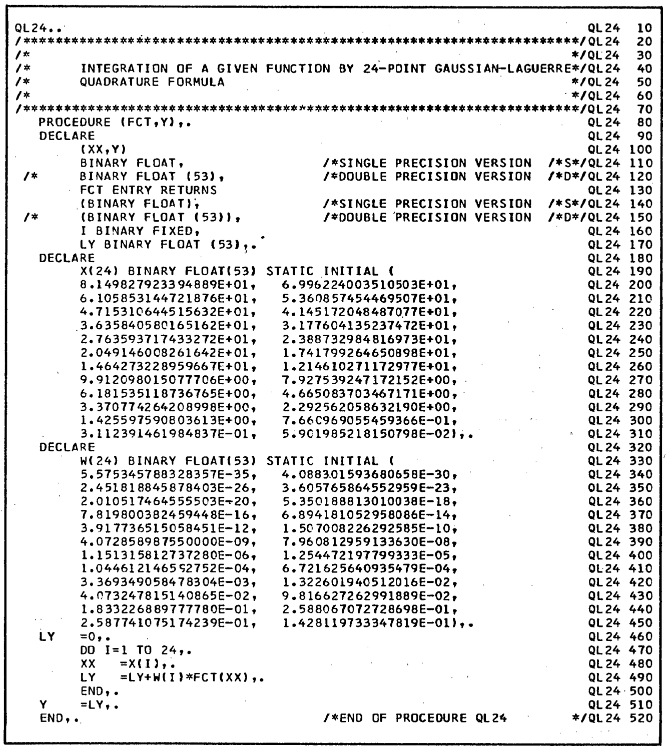
```
QL4..
/**********************************************************************/
/*                                                                    */
/*      INTEGRATION OF A GIVEN FUNCTION BY 4-POINT GAUSSIAN-LAGUERRE */
/*      QUADRATURE FORMULA                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (X,Y)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
   X    =9.395070912301133E+00,.
   Y    =5.392947055613275E-04*FCT(X),.
   X    =4.536620296921128E+00,.
   Y    =Y+3.888790851500538E-02*FCT(X),.
   X    =1.745761101158347E+00,.
   Y    =Y+3.574186924377997E-01*FCT(X),.
   X    =3.225476896193923E-01,.
   Y    =Y+6.031541043416336E-01*FCT(X),.
   END,.                               /*END OF PROCEDURE QL4        */
```

```
QL8..
/**********************************************************************/
/*                                                                    */
/*      INTEGRATION OF A GIVEN FUNCTION BY 8-POINT GAUSSIAN-LAGUERRE */
/*      QUADRATURE FORMULA                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),
        X(16) BINARY FLOAT (53) STATIC INITIAL
        (2.286313173688926E+01,     1.048001174871510E-09,
        1.574067864127800E+01,     8.485746716272532E-07,
        1.075851601018100E+01,     9.076508773358213E-05,
        7.045905402393466E+00,     2.794536235225673E-03,
        4.266700170287659E+00,     3.334349226121565E-02,
        2.251086629866131E+00,     1.757949866371718E-01,
        9.037017767993799E-01,     4.187867808143430E-01,
        1.7C2796323C51010E-01,     3.691885893416375E-01),.
   LY   =0,.
        DO I=1 TO 15 BY 2,.
        XX   =X(I),.
        LY   =LY+X(I+1)*FCT(XX),.
        END,.
   Y    =LY,.
   END,.                               /*END OF PROCEDURE QL8        */
```

```
QL12..
/**********************************************************************/
/*                                                                    */
/*      INTEGRATION OF A GIVEN FUNCTION BY 12-POINT GAUSSIAN-LAGUERRE*/
/*      QUADRATURE FORMULA                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),
        X(24) BINARY FLOAT (53) STATIC INITIAL
        (3.709912104446692E+01,     8.148077467426242E-16,
        2.848796725C98400E+01,     3.C61601635035021E-12,
        2.215109C37939701E+01,     1.342391030515004E-09,
        1.711685518746226E+01,     1.668493376540910E-07,
        1.300605499330635E+01,     8.365055856819799E-06,
        9.621316842456867E+00,     2.C32315926629994E-04,
        6.844525453115177E+00,     2.663973541865316E-03,
        4.599227639418348E+00,     2.C1C238115463410E-02,
        2.83375133774350 7E+00,    9.C44922221168093E-02,
        1.51261C269776419E+00,     2.44C820113198776E-01,
        6.117574845151307E-01,     3.777592758731380E-01,
        1.157221173580207E-01,     2.647313710554432E-01),.
   LY   =0,.
        DO I=1 TO 23 BY 2,.
        XX   =X(I),.
        LY   =LY+X(I+1)*FCT(XX),.
        END,.
   Y    =LY,.
   END,.                               /*END OF PROCEDURE QL12       */
```



```
QL16..
/**********************************************************************/
/*                                                                    */
/*      INTEGRATION OF A GIVEN FUNCTION BY 16-POINT GAUSSIAN-LAGUERRE*/
/*      QUADRATURE FORMULA                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),               /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),          /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),            /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),
        X(32) BINARY FLOAT (53) STATIC INITIAL
        (5.170116033954332E+01,    4.161462370372855E-22,
        4.194045264768833E+01,    5.050473700035513E-18,
        3.458339870228663E+01,    6.297967002517868E-15,
        2.857872974288214E+01,    2.127079033224103E-12,
        2.351590569399191E+01,    2.862350242973882E-10,
        1.918015685675313E+01,    1.881024841079673E-08,
        1.544152736878162E+01,    6.828319330871200E-07,
        1.221422336886616E+01,    1.484458687398130E-05,
        9.438314336391939E+00,    2.042719153082785E-04,
        7.070338535048234E+00,    1.849070943526311E-03,
        5.078018614549768E+00,    1.129990008033945E-02,
        3.437086633893207E+00,    4.732892869412522E-02,
        2.129283645098381E+00,    1.362969342963775E-01,
        1.141057774831227E+00,    2.657957776442142E-01,
        4.626963289150808E-01,    3.310578549508842E-01,
        8.764941047892784E-02,    2.061517149578010E-01),.
   LY   =0,.
        DO I=1 TO 31 BY 2,.
        XX   =X(I),.
        LY   =LY+X(I+1)*FCT(XX),.
        END,.
   Y    =LY,.
   END,.                              /*END OF PROCEDURE QL16        */
```



```
QL24..
/**********************************************************************/
/*                                                                    */
/*      INTEGRATION OF A GIVEN FUNCTION BY 24-POINT GAUSSIAN-LAGUERRE*/
/*      QUADRATURE FORMULA                                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        (XX,Y)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),            /*DOUBLE PRECISION VERSION  /*D*/
        FCT ENTRY RETURNS
        (BINARY FLOAT),               /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),          /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),.
   DECLARE
        X(24) BINARY FLOAT(53) STATIC INITIAL (
        8.149827923394889E+01,   6.996224003510503E+01,
        6.105853144721876E+01,   5.360857454469507E+01,
        4.715310644515632E+01,   4.145172048487077E+01,
        3.635840580165162E+01,   3.177604135237472E+01,
        2.763593717433272E+01,   2.388732984816973E+01,
        2.049146008261642E+01,   1.741799264650898E+01,
        1.464273228959667E+01,   1.214610271172977E+01,
        9.912098015077706E+00,   7.927539247172152E+00,
        6.181535118736765E+00,   4.665083703467171E+00,
        3.370774264208998E+00,   2.292562058632190E+00,
        1.425597590803613E+00,   7.660969055459366E-01,
        3.112391461984837E-01,   5.901985218150798E-02),.
   DECLARE
        W(24) BINARY FLOAT(53) STATIC INITIAL (
        5.575345788328357E-35,   4.088301593680658E-30,
        2.451818845878403E-26,   3.605765864552959E-23,
        2.010517464555503E-20,   5.350188813010038E-18,
        7.819800382459448E-16,   6.894181052958086E-14,
        3.917736515058451E-12,   1.507008226292585E-10,
        4.072858987550000E-09,   7.960812959133630E-08,
        1.151315812737280E-06,   1.254472197799333E-05,
        1.044612146592752E-04,   6.721625640935479E-04,
        3.369349058478304E-03,   1.322601940512016E-02,
        4.073247815140865E-02,   9.816627262991889E-02,
        1.833226889777780E-01,   2.588067072728698E-01,
        2.587741075174239E-01,   1.428119733347819E-01),.
   LY   =0,.
        DO I=1 TO 24,.
        XX   =X(I),.
        LY   =LY+W(I)*FCT(XX),.
        END,.
   Y    =LY,.
   END,.                              /*END OF PROCEDURE QL24        */
```

Purpose:

QLn computes the integral value $Y = \int_0^\infty e^{-X} FCT(X)dX$ for a given function FCT(X), by Gaussian-Laguerre quadrature formulas.

Usage:

CALL QLn (FCT, Y);

FCT - ENTRY
Given procedure for the computation of the function values.
This procedure must be supplied by the user.

Usage:
FCT(X)
FCT(X) - BINARY FLOAT [(53)]
Resultant function value.
X - BINARY FLOAT [(53)]
Given argument value.

Y - BINARY FLOAT [(53)]
Resultant integral value.

Remarks:

The n in the name QLn indicates the number of nodes used for the calculation of Y.

Method:

Quadrature formulas of Gauss-Laguerre are used for the evaluation of the integral values.

For reference see:

H. E. Salzer, R. Zucker, "Table of Zeros and Weight Factors of the First Fifteen Laguerre Polynomials", Bul. Amer. Math. Soc., vol. 55 (1949), pp. 1004-1012.

V. I. Krylow, Approximate Calculation of Integrals, Macmillan, New York-London, 1962, pp 130-132 and 347-352.

Shao, Chen, Frank, "Tables of Zeros and Gaussian Weights of Certain Associated Laguerre Polynomials and the Related Generalized Hermite Polynomials", IBM Technical Report TR 00.1100, March 1964, pp. 24-25.

Mathematical Background:

Formulas of Gauss- Laguerre are used to compute

$$y = \int_0^\infty e^{-x} f(x)\, dx$$

Let n denote the number of nodes used for the calculation of the integral value y. The value y is approximated by a weighted sum of function values:

$$y^{(n)} = \sum_{k=1}^{n} [A_k^{(n)} \cdot f(x_k^{(n)})]$$

The value $y^{(n)}$ is exact whenever f(x) is a polynomial of degree less than or equal to 2n-1. The nodes $x_k^{(n)}$ are the roots of the Laguerre polynomials $L_n(x)$ of degree n.

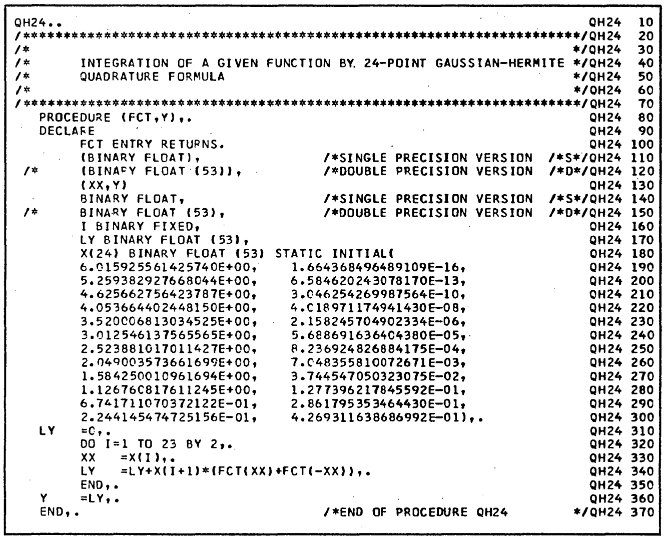
- Subroutine QHn (n = 2, 4, 8, 16, 24, 32, 48)

```
QH2..                                                                   QH2
/**********************************************************************/QH2
/*                                                                    */QH2
/*      INTEGRATICN OF A GIVEN FUNCTION BY 2-POINT GAUSSIAN-HERMITE   */QH2
/*      QUADPATURE FOPMULA                                            */QH2
/*                                                                    */QH2
/**********************************************************************/QH2
   PFOCEDUPE (FCT,Y),.                                                  QH2
   DECLARE                                                              QH2
        FCT ENTRY RETURNS •                                             QH2
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/QH2
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/QH2
        (X,Y,Z)                                                         QH2
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/QH2
 /*     BINARY FLOAT(53),.             /*DOUBLE PRECISION VERSION  /*D*/QH2
   X    =7.071C67811865475E-01,.                                        QH2
   Z    =-X,.                                                           QH2
   Y    =8.862269254527580E-01*(FCT(X)+FCT(Z)),.                        QH2
   END,.                               /*END OF PROCEDURE QH2         */QH2
```

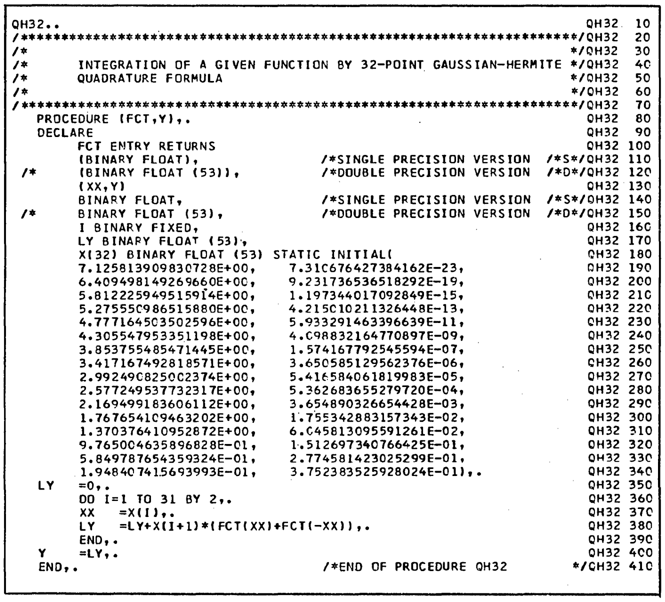
```
QH4..                                                                   QH4
/**********************************************************************/QH4
/*                                                                    */QH4
/*      INTEGRATION OF A GIVEN FUNCTION BY 4-POINT GAUSSIAN-HERMITE   */QH4
/*      QUADRATURE FORMULA                                            */QH4
/*                                                                    */QH4
/**********************************************************************/QH4
   PROCEDURE (FCT,Y),.                                                  QH4
   DECLARE                                                              QH4
        FCT ENTRY RETUFNS                                               QH4
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/QH4
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/QH4
        W BINARY FLOAT(53),                                             QH4
        (X,Y,Z)                                                         QH4
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/QH4
 /*     BINARY FLOAT(53),.             /*DOUBLE PRECISION VERSION  /*D*/QH4
   X    =1.650680123885785E+00,.                                        QH4
   Z    =-X,.                                                           QH4
   W    =8.131283544724518E-02*(FCT(X)+FCT(Z)),.                        QH4
   X    =5.246476232752903E-01,.                                        QH4
   Z    =-X,.                                                           QH4
   Y    =W+8.049140900055128E-01*(FCT(X)+FCT(Z)),.                      QH4
   END,.                               /*END OF PROCEDURE QH4         */QH4
```

```
QH8..                                                                   QH8
/**********************************************************************/QH8
/*                                                                    */QH8
/*      INTEGRATION OF A GIVEN FUNCTION BY 8-POINT GAUSSIAN-HERMITE   */QH8
/*      QUADPATUFE FORMULA                                            */QH8
/*                                                                    */QH8
/**********************************************************************/QH8
   PROCEDUPE (FCT,Y),.                                                  QH8
   DECLARE                                                              QH8
        FCT ENTRY RETUPNS                                               QH8
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/QH8
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/QH8
        (XX,Y)                                                          QH8
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/QH8
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/QH8
        I BINARY FIXED,                                                 QH8
        LY BINARY FLOAT (53),                                           QH8
        X( 8) BINARY FLOAT (53) STATIC INITIAL(                         QH8
        2.930637420257244E+0C,     1.996040722113676E-04,               QH8
        1.981656756695843E+00,     1.707798300741348E-02,               QH8
        1.157193712446780E+00,     2.078023258148919E-01,               QH8
        3.811869902073221E-01,     6.611470125582413E-01),.             QH8
   LY   =0,.                                                            QH8
        DO I=1 TO  7 BY 2,.                                             QH8
        XX   =X(I),.                                                    QH8
        LY   =LY+X(I+1)*(FCT(XX)+FCT(-XX)),.                            QH8
        END,.                                                           QH8
   Y    =LY,.                                                           QH8
   END,.                               /*END OF PROCEDURE QH8         */QH8
```

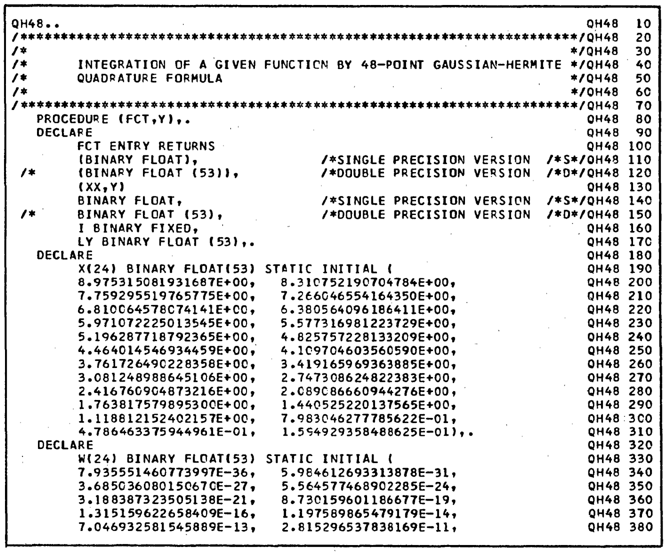
```
QH16..                                                                 QH16
/*********************************************************************/QH16
/*                                                                   */QH16
/*      INTEGRATION OF A GIVEN FUNCTION BY 16-POINT GAUSSIAN-HERMITE */QH16
/*      QUADRATURE FORMULA                                           */QH16
/*                                                                   */QH16
/*********************************************************************/QH16
   PROCEDURE (FCT,Y),.                                                 QH16
   DECLARE                                                             QH16
        FCT ENTRY RETURNS                                              QH16
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION /*S*/QH16
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION /*D*/QH16
        (XX,Y)                                                         QH16
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION /*S*/QH16
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION /*D*/QH16
        I BINARY FIXED,                                                QH16
        LY BINARY FLOAT (53),                                          QH16
        X(16) BINARY FLOAT (53) STATIC INITIAL(                        QH16
        4.688738939305818E+00,     2.654807474011182E-10,              QH16
        3.869447904860123E+00,     2.320980844865211E-07,              QH16
        3.176999161979956E+00,     2.711860092537882E-05,              QH16
        2.546202157847481E+00,     9.322840086241805E-04,              QH16
        1.951787990916254E+00,     1.288031153550997E-02,              QH16
        1.380258539198881E+00,     8.381004139898583E-02,              QH16
        8.229514491446559E-01,     2.806474585285337E-01,              QH16
        2.734810461381525E-01,     5.079294790166137E-01),.            QH16
   LY   =0,.                                                           QH16
        DO I=1 TO 15 BY 2,.                                            QH16
        XX   =X(I),.                                                   QH16
        LY   =LY+X(I+1)*(FCT(XX)+FCT(-XX)),.                           QH16
        END,.                                                          QH16
   Y    =LY,.                                                          QH16
   END,.                               /*END OF PROCEDURE QH16       */QH16
```



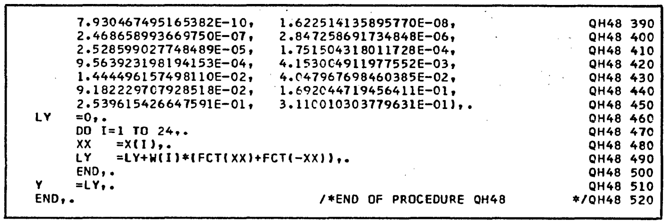
```
QH24..
/***********************************************************************/
/*                                                                     */
/*      INTEGRATION OF A GIVEN FUNCTION BY. 24-POINT GAUSSIAN-HERMITE */
/*      QUADRATURE FORMULA                                             */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS.
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINAFY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),
        X(24) BINARY FLOAT (53) STATIC INITIAL(
        6.015925561425740E+00,   1.664368496489109E-16,
        5.259382927668044E+00,   6.584620243078170E-13,
        4.625662756423787E+00,   3.C46254269987564E-10,
        4.053664402448150E+00,   4.C18971174941430E-08,
        3.520C06813034525E+00,   2.158245704902334E-06,
        3.012546137565565E+00,   5.688691636404380E-05,
        2.523881017011427E+00,   8.236924826884175E-04,
        2.049003573661699E+00,   7.C48355810072671E-03,
        1.584250010961694E+00,   3.744547050323075E-02,
        1.126760817611245E+00,   1.277396217845592E-01,
        6.741711070372122E-01,   2.861795353464430E-01,
        2.244145474725156E-01,   4.269311638686992E-01),.
  LY    =C,.
        DO I=1 TO 23 BY 2,.
        XX    =X(I),.
        LY    =LY+X(I+1)*(FCT(XX)+FCT(-XX)),.
        END,.
  Y     =LY,.
  END,.                                /*END OF PROCEDURE QH24        */
```



```
QH32..
/***********************************************************************/
/*                                                                     */
/*      INTEGRATION OF A GIVEN FUNCTION BY 32-POINT GAUSSIAN-HERMITE */
/*      QUADRATURE FORMULA                                             */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),
        X(32) BINARY FLOAT (53) STATIC INITIAL(
        7.125813909830728E+00,   7.31C676427384162E-23,
        6.409498149269660E+00,   9.231736536518292E-19,
        5.812225949515914E+00,   1.197344017092849E-15,
        5.275550986515880E+00,   4.215C10211326448E-13,
        4.777164503502596E+00,   5.933291463396639E-11,
        4.305547953351198E+00,   4.C98832164770897E-09,
        3.853755485471445E+00,   1.574167792545594E-07,
        3.417167492818571E+00,   3.650585129562376E-06,
        2.992490825002374E+00,   5.416584061819983E-05,
        2.577249537732317E+00,   5.362683655279720E-04,
        2.169499183606112E+00,   3.654890326654428E-03,
        1.767654109463202E+00,   1.755342883157343E-02,
        1.370376410952872E+00,   6.C45813095591261E-02,
        9.765004635896828E-01,   1.512697340766425E-01,
        5.849787654359324E-01,   2.774581423025299E-01,
        1.948407415693993E-01,   3.752383525928024E-01),.
  LY    =0,.
        DO I=1 TO 31 BY 2,.
        XX    =X(I),.
        LY    =LY+X(I+1)*(FCT(XX)+FCT(-XX)),.
        END,.
  Y     =LY,.
  END,.                                /*END OF PROCEDURE QH32        */
```



```
QH48..
/***********************************************************************/
/*                                                                     */
/*      INTEGRATION OF A GIVEN FUNCTICN BY 48-POINT GAUSSIAN-HERMITE */
/*      QUADRATURE FORMULA                                             */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        I BINARY FIXED,
        LY BINARY FLOAT (53),.
   DECLARE
        X(24) BINARY FLOAT(53) STATIC INITIAL (
        8.975315081931687E+00,  8.310752190704784E+00,
        7.759295519765775E+00,  7.266046554164350E+00,
        6.810064578074141E+00,  6.380564096186411E+00,
        5.971072225013545E+00,  5.577316981223729E+00,
        5.196287718792365E+00,  4.825757228133209E+00,
        4.464014546934459E+00,  4.109704603560590E+00,
        3.761726490228358E+00,  3.419165969363885E+00,
        3.081248988645106E+00,  2.747308624822383E+00,
        2.416760904873216E+00,  2.089086660944276E+00,
        1.763817579895300E+00,  1.440525220137565E+00,
        1.118812152402157E+00,  7.983046277785622E-01,
        4.786463375944961E-01,  1.594929358488625E-01),.
   DECLARE
        W(24) BINARY FLOAT(53) STATIC INITIAL (
        7.935551460773997E-36,  5.984612693313878E-31,
        3.685036080150670E-27,  5.564577468902285E-24,
        3.188387323505138E-21,  8.730159601186677E-19,
        1.315159622658409E-16,  1.197589865479179E-14,
        7.046932581545889E-13,  2.815296537838169E-11,
```



```
        7.930467495165382E-10,  1.622514135895770E-08,
        2.468658993669750E-07,  2.847258691734848E-06,
        2.528599027748489E-05,  1.751504318011728E-04,
        9.563923198194153E-04,  4.153004911977552E-03,
        1.444496157498110E-02,  4.047967698460385E-02,
        9.182229707928518E-02,  1.692044719456411E-01,
        2.539615426647591E-01,  3.110010303779631E-01),.
  LY    =0,.
        DO I=1 TO 24,.
        XX    =X(I),.
        LY    =LY+W(I)*(FCT(XX)+FCT(-XX)),.
        END,.
  Y     =LY,.
  END,.                                /*END OF PROCEDURE QH48        */
```

Purpose:

QHn computes the integral value $Y=\int_{-\infty}^{+\infty} e^{-X^2}\ FCT(X)\,dX$ for a given function FCT(X), using Gaussian-Hermite quadrature formulas.

Usage:

CALL QHn (FCT, Y);

FCT - ENTRY
Given procedure for the computation of the function values, which must be supplied by the user.

Usage
FCT(X);
FCT(X) - BINARY FLOAT [(53)]
Resultant function value.
X - BINARY FLOAT [(53)]
Given argument value.

Y - BINARY FLOAT [(53)]
Resultant integral value.

Remarks:

The number n in the name QHn indicates the number of nodes used for the calculation of Y.

In case of an even function $f(x) = \varphi(x^2)$, f(x) may be changed by means of the transformation $t = x^2$ into:

$$y = \int_0^\infty \frac{e^{-t}\varphi(t)}{\sqrt{t}}\,dt$$

This is a form suitable to subroutines QAn, the use of which saves approximately half of the computation time.

Method:

Quadrature formulas of Gauss-Hermite are used for the computation of the integral values.

For reference see:

H. E. Salzer, R. Zucker, R. Capuano, Table of Zeros and Weight Factors of the First Twenty

Hermite Polynomials, F. Res. Nat. Bur. Standards, vol. 48 (1952), pp. 111-116.

V. I. Krylow, Approximate Calculation of Integrals, Macmillan, New York-London, 1962, pp. 129-130 and 343-346.

Mathematical Background:

Quadrature formulas of Gauss-Hermite are used to compute

$$y = \int_{-\infty}^{+\infty} e^{-x^2} f(x)\, dx$$

Let n denote the number of nodes used for the calculation of the integral value y. The value y is approximated by a weighted sum of function values:

$$y^{(n)} = \sum_{k=1}^{n} A_k^{(n)} f(x_k^{(n)})$$

The value $y^{(n)}$ is exact whenever f(x) is a polynomial of degree less than or equal to 2n-1.

The nodes $x_k^{(n)}$ are the roots of the Hermite polynomials $H_n(x)$ of degree n.

The weights $A_k^{(n)}$ and the nodes $x_k^{(n)}$ are symmetric with respect to the origin x=0:

$$A_k^{(n)} = A_{n-k+1}^{(n)}, \quad x_k^{(n)} = - x_{n-k+1}^{(n)}$$

⊙ Subroutine QAn (n = 2, 4, 8, 12, 16, 24)

```
QA2..                                                                   QA2
/**********************************************************************/QA2
/*                                                                    */QA2
/*      INTEGRATION OF A GIVEN FUNCTION BY ASSOCIATED 2-POINT         */QA2
/*      GAUSSIAN-LAGUERRE QUADRATURE FORMULA                          */QA2
/*                                                                    */QA2
/**********************************************************************/QA2
   PROCEDURE (FCT,Y),.                                                  QA2
   DECLARE                                                              QA2
         FCT ENTRY RETURNS                                              QA2
         (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/QA2
 /*      (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/QA2
         (X,Y)                                                          QA2
         BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/QA2
 /*      BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/QA2
   X     =2.724744871391589E+00,.                                       QA2
   Y     =1.626256708944903E-01*FCT(X),.                                QA2
   X     =2.752551286084109E-01,.                                       QA2
   Y     =Y+1.609828180011026E+00*FCT(X),.                              QA2
   END,.                                /*END OF PROCEDURE QA2         */QA2
```

```
QA4..                                                                   QA4
/**********************************************************************/QA4
/*                                                                    */QA4
/*      INTEGRATION OF A GIVEN FUNCTION BY ASSOCIATED 4-POINT         */QA4
/*      GAUSSIAN-LAGUERRE QUADRATURE FORMULA                          */QA4
/*                                                                    */QA4
/**********************************************************************/QA4
   PROCEDURE (FCT,Y),.                                                  QA4
   DECLARE                                                              QA4
         FCT ENTRY FETUPNS.                                             QA4
         (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/QA4
 /*      (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/QA4
         (X,Y)                                                          QA4
         BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/QA4
 /*      BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/QA4
   X     =8.588635689012034E+00,.                                       QA4
   Y     =3.992081444227352E-04*FCT(X),.                                QA4
   X     =3.926963501358287E+00,.                                       QA4
   Y     =Y+3.415596601482695E-02*FCT(X),.                              QA4
   X     =1.339097288126361E+00,.                                       QA4
   Y     =Y+4.156046516297838E-01*FCT(X),.                              QA4
   X     =1.453035215033171E-01,.                                       QA4
   Y     =Y+1.322294025116483E+00*FCT(X),.                              QA4
   END,.                                /*END OF PROCEDURE QA4         */QA4
```

```
QA8..                                                                   QA8
/**********************************************************************/QA8
/*                                                                    */QA8
/*      INTEGRATION OF A GIVEN FUNCTION BY ASSOCIATED 8-POINT         */QA8
/*      GAUSSIAN-LAGUERRE QUADRATURE FORMULA                          */QA8
/*                                                                    */QA8
/**********************************************************************/QA8
   PROCEDURE (FCT,Y),.                                                  QA8
   DECLARE                                                              QA8
         FCT ENTRY RETURNS                                              QA8
         (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/QA8
 /*      (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/QA8
         (XX,Y)                                                         QA8
         BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/QA8
 /*      BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/QA8
         LY BINARY FLOAT (53),                                          QA8
         I BINARY FIXED,                                                QA8
         X(16) BINARY FLOAT (53) STATIC INITIAL                         QA8
         (2.198427284096265E+01,    5.309614948022364E-10,              QA8
         1.497262708842639E+01,    4.641961689730421E-07,               QA8
         1.009332367522134E+01,    5.423720185075763E-05,               QA8
         6.483145428627170E+00,    1.864568017248361E-03,               QA8
         3.809476361484907E+00,    2.576062307101995E-02,               QA8
         1.905113635031428E+00,    1.676200827979717E-01,               QA8
         6.772490876492892E-01,    5.612949170570674E-01,               QA8
         7.479188259681827E-02,    1.015858958033227E+00),.             QA8
   LY    =0,.                                                           QA8
         DO I=1 TO 15 BY 2,.                                            QA8
         XX    =X(I),.                                                  QA8
         LY    =LY+X(I+1)*FCT(XX),.                                     QA8
         END,.                                                          QA8
   Y     =LY,.                                                          QA8
   END,.                                /*END OF PROCEDURE QA8         */QA8
```

```
QA12..                                                                 QA12
/*********************************************************************/QA12
/*                                                                   */QA12
/*      INTEGRATION OF A GIVEN FUNCTION BY ASSOCIATED 12-POINT       */QA12
/*      GAUSSIAN-LAGUERRE QUADRATURE FORMULA                         */QA12
/*                                                                   */QA12
/*********************************************************************/QA12
   PROCEDURE (FCT,Y),.                                                 QA12
   DECLARE                                                             QA12
         FCT ENTRY RETURNS                                             QA12
         (BINARY FLOAT),                /*SINGLE PRECISION VERSION /*S*/QA12
 /*      (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION /*D*/QA12
         (XX,Y)                                                        QA12
         BINARY FLOAT,                  /*SINGLE PRECISION VERSION /*S*/QA12
 /*      .BINARY FLOAT (53),            /*DOUBLE PRECISION VERSION /*D*/QA12
         LY BINARY FLOAT (53),                                         QA12
         I BINARY FIXED,                                               QA12
         X(24) BINARY FLOAT (53) STATIC INITIAL                        QA12
         (3.619136036061560E+01,    3.328736992978218E-16,             QA12
         2.766110877984609E+01,    1.316924048615634E-12,              QA12
         2.139675593616611E+01,    6.092508539975128E-10,              QA12
         1.643219508767531E+01,    8.037942349882859E-08,              QA12
         1.239044796380947E+01,    4.316491409804667E-06,              QA12
         9.075434230961203E+00,    1.137738327280876E-04,              QA12
         6.369975388030635E+00,    1.647384965376835E-03,              QA12
         4.198415644878413E+00,    1.409671162014534E-02,              QA12
         2.509848097232128E+00,    7.489094100646149E-02,              QA12
         1.269589940103961E+00,    2.554792435691183E-01,              QA12
         4.545066815637803E-01,    5.723590706928860E-01,              QA12
         5.036188911729395E-02,    8.538623277373985E-01),.            QA12
   LY    =0,.                                                          QA12
         DO I=1 TO 23 BY 2,.                                           QA12
         XX    =X(I),.                                                 QA12
         LY    =LY+X(I+1)*FCT(XX),.                                    QA12
         END,.                                                         QA12
   Y     =LY,.                                                         QA12
   END,.                                /*END OF PROCEDURE QA12       */QA12
```

```
QA16..
/***********************************************************************/
/*                                                                     */
/*     INTEGRATION OF A GIVEN FUNCTION BY ASSOCIATED 16-POINT          */
/*     GAUSSIAN-LAGUERRE QUADRATURE FORMULA                            */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (FCT,Y),.
   DECLARE
        FCT ENTRY RETURNS
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        LY BINARY FLOAT (53),
        I BINARY FIXED,
        X(32) BINARY FLOAT (53) STATIC INITIAL
        (5.C777223877537C8E+01,    1.462135285476832E-22,
        4.10816665254912CE+01,    1.846347307303658E-18,
        3.378197048822617E+01,    2.394688034185697E-15,
        2.783143821132868E+01,    8.430020422652895E-13,
        2.282130069352521E+01,    1.186658292679328E-10,
        1.85377431786C669E+01,    8.197664329541793E-09,
        1.485143134180125E+01,    3.148335585091188E-07,
        1.167703367397596E+01,    7.301170259124752E-06,
        8.955001337723390E+00,    1.083316812363997E-04,
        6.642215179741444E+00,    1.072536731055944E-03,
        4.706726707667587E+00,    7.309780653308856E-03,
        3.124601050702144E+00,    3.510685766314686E-02,
        1.8779315C7696074E+00,    1.209162619118252E-01,
        9.535531553908655E-01,    3.025394681532850E-01,
        3.422001560109477E-01,    5.549162846050598E-01,
        3.796291457531345E-02,    7.504767051856048E-01),.
  LY    =0,.
        DO I=1 TO 31 BY 2,.
        XX    =X(I),.
        LY    =LY+X(I+1)*FCT(XX),.
        END,.
  Y     =LY,.
  END,.                              /*END OF PROCEDURE QA16       */
```

```
QA24..
/***********************************************************************/
/*                                                                     */
/*     INTEGRATION OF A GIVEN FUNCTION BY ASSOCIATED 24-POINT          */
/*     GAUSSIAN-LAGUEPPE QUADRATURE FORMULA                            */
/*                                                                     */
/***********************************************************************/
   PROCEDUPE (FCT,Y),.
   DECLARE
        FCT ENTRY RETUPNS·
        (BINARY FLOAT),                /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT (53)),           /*DOUBLE PRECISION VERSION  /*D*/
        (XX,Y)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        LY BINARY FLOAT (53),
        I BINARY FIXED,.
   DECLARE
        X(24) BINARY FLOAT(53) STATIC INITIAL (
        8.055628C81995C41E+01,   6.906860197530437E+01,
        6.020666696305722E+01,   5.279543252728363E+01,
        4.637697955754013E+01,   4.071159818554311E+01,
        3.565370351632821E+01,   3.110646470904657E+01,
        2.7C01406C5647236E+01,   2.328793282487992E+01,
        1.992742587524246E+01,   1.688967192852711E+01,
        1.415058618728576E+01,   1.169069592605607E+01,
        9.494095330C26488E+00,   7.547704680023454E+00,
        5.840733271323608E+0C,   4.364283076935306E+00,
        3.111C52455147713E+00,   2.075112909852381E+00,
        1.251740632362746E+0C,   6.372902787326688E-01,
        2.291C23164926243E-01,   2.543799658568936E-02),.
   DECLAPE
        W(24) BINARY FLOAT(53) STATIC INITIAL (
        1.587110292154799E-35,   1.196922538662776E-30,
        7.370072160301340E-27,   1.112915493780457E-23,
        6.376774647010277E-21,   1.746031920237335E-18,
        2.63C3192453168 17E-16,   2.395179730958359E-14,
        1.4C9386516309178E-12,   5.630593075676338E-11,
        1.586093499C33076E-09,   3.245028271791540E-08,
        4.9373179873395C1E-07,   5.694517383469696E-06,
        5.057198C55496978E-05,   3.5C3C08636023457E-04,
        1.912784639638831E-03,   8.3C6C09823955105E-03,
        2.888992314996220E-02,   8.095935396920770E-02,
        1.836445941585704E-01,   3.384089438912822E-C1,
        5.C79230853295182E-01,   6.220020607559262E-C1),.
  LY    =C,.
        DO I=1 TO 24,.
        XX    =X(I),.
        LY    =LY+W(I)*FCT(XX),.
        END,.
  Y     =LY,.
  END,.                              /*END OF PROCEDURE QA24       */
```

Purpose:

QAn computes the integral value $Y=\int_0^\infty \frac{e^{-X}\ FCT(X)}{\sqrt{X}}\,dX$ for a given function FCT(X), using associated Gaussian-Laguerre quadrature formulas.

Usage:

CALL QAn (FCT, Y);

FCT - ENTRY
Given procedure for the computation of the function values. This procedure must be supplied by the user.

Usage
FCT(X);
FCT(X) - BINARY FLOAT [(53)]
Resultant function value.
X - BINARY FLOAT [(53)]
Given argument value.

Y - BINARY FLOAT [(53)]
Resultant integral value.

Remarks:

The n in the name QAn indicates the number of nodes used for the calculation of Y.

Method:

Quadrature formulas of Gauss- Laguerre are used for the evaluation of the integral value.

For reference see:

Concus, Cassatt, Jaehnig, Melby, "Tables for the Evaluation of $\int_0^\infty x^\beta e^{-x} f(x)\,dx$ by Gauss-Laguerre Quadrature, MTAC, vol. 17, No. 83 (1963), pp 245-256.

Shao, Chen, Frank, "Tables of Zeros and Gaussian Weights of Certain Associated Laguerre Polynomials and the Related Generalized Hermite Polynomials", IBM Technical Report TR 00.1100, March 1964, pp. 15-16.

Mathematical Background:

Formulas of Gauss-Laguerre are used to compute

$$y = \int_0^\infty \frac{e^{-x}\,f(x)}{\sqrt{x}}\,dx$$

Let n denote the number of nodes used for the calculation of the integral value y.

The value y is approximated by a weighted sum of function values:

$$y^{(n)} = \sum_{k=1}^{n} [A_k^{(n)}\ f\,(x_k^{(n)}\,)\ ]$$

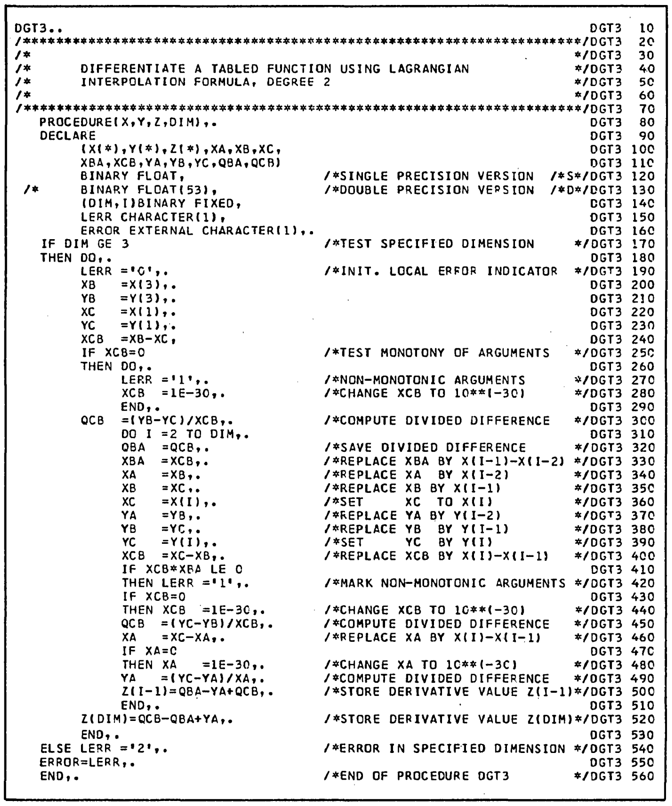
The value $y^{(n)}$ is exact whenever f(x) is a polynomial of degree less than or equal to 2n-1.

The nodes $x_k^{(n)}$ are the roots of the associated Laguerre polynomials $L_n^{(-1/2)}(x)$ of degree n.

Numerical Differentiation

Differentiation of Tabulated Functions

- Subroutine DGT3



```
DGT3..                                                                  DGT3  10
/**********************************************************************/DGT3  20
/*                                                                    */DGT3  30
/*      DIFFERENTIATE A TABLED FUNCTION USING LAGRANGIAN              */DGT3  40
/*      INTERPOLATION FORMULA, DEGREE 2                               */DGT3  50
/*                                                                    */DGT3  60
/**********************************************************************/DGT3  70
   PROCEDURE(X,Y,Z,DIM),.                                               DGT3  80
   DECLARE                                                              DGT3  90
         (X(*),Y(*),Z(*),XA,XB,XC,                                      DGT3 100
         XBA,XCB,YA,YB,YC,QBA,QCB)                                      DGT3 110
         BINARY FLOAT,                /*SINGLE PRECISION VERSION  /*S*/DGT3 120
 /*      BINARY FLOAT(53),            /*DOUBLE PRECISION VERSION  /*D*/DGT3 130
         (DIM,I)BINARY FIXED,                                           DGT3 140
         LERR CHARACTER(1),                                             DGT3 150
         ERROR EXTERNAL CHARACTER(1),.                                  DGT3 160
   IF DIM GE 3                        /*TEST SPECIFIED DIMENSION     */DGT3 170
   THEN DO,.                                                            DGT3 180
         LERR ='0',.                  /*INIT. LOCAL ERROR INDICATOR  */DGT3 190
         XB   =X(3),.                                                   DGT3 200
         YB   =Y(3),.                                                   DGT3 210
         XC   =X(1),.                                                   DGT3 220
         YC   =Y(1),.                                                   DGT3 230
         XCB  =XB-XC,                                                   DGT3 240
         IF XCB=0                     /*TEST MONOTONY OF ARGUMENTS   */DGT3 250
         THEN DO,.                                                      DGT3 260
               LERR ='1',.            /*NON-MONOTONIC ARGUMENTS      */DGT3 270
               XCB  =1E-30,.          /*CHANGE XCB TO 10**(-30)      */DGT3 280
               END,.                                                    DGT3 290
         QCB  =(YB-YC)/XCB,.          /*COMPUTE DIVIDED DIFFERENCE   */DGT3 300
               DO I =2 TO DIM,.                                         DGT3 310
               QBA  =QCB,.            /*SAVE DIVIDED DIFFERENCE      */DGT3 320
               XBA  =XCB,.            /*REPLACE XBA BY X(I-1)-X(I-2) */DGT3 330
               XA   =XB,.             /*REPLACE XA  BY X(I-2)        */DGT3 340
               XB   =XC,.             /*REPLACE XB BY X(I-1)         */DGT3 350
               XC   =X(I),.           /*SET     XC  TO X(I)          */DGT3 360
               YA   =YB,.             /*REPLACE YA BY Y(I-2)         */DGT3 370
               YB   =YC,.             /*REPLACE YB  BY Y(I-1)        */DGT3 380
               YC   =Y(I),.           /*SET     YC  BY Y(I)          */DGT3 390
               XCB  =XC-XB,.          /*REPLACE XCB BY X(I)-X(I-1)   */DGT3 400
               IF XCB*XBA LE 0                                          DGT3 410
               THEN LERR ='1',.       /*MARK NON-MONOTONIC ARGUMENTS */DGT3 420
               IF XCB=0                                                 DGT3 430
               THEN XCB  =1E-30,.     /*CHANGE XCB TO 10**(-30)      */DGT3 440
               QCB  =(YC-YB)/XCB,.    /*COMPUTE DIVIDED DIFFERENCE   */DGT3 450
               XA   =XC-XA,.          /*REPLACE XA BY X(I)-X(I-1)    */DGT3 460
               IF XA=0                                                  DGT3 470
               THEN XA   =1E-30,.     /*CHANGE XA TO 10**(-30)       */DGT3 480
               YA   =(YC-YA)/XA,.     /*COMPUTE DIVIDED DIFFERENCE   */DGT3 490
               Z(I-1)=QBA-YA+QCB,.    /*STORE DERIVATIVE VALUE Z(I-1)*/DGT3 500
               END,.                                                    DGT3 510
         Z(DIM)=QCB-QBA+YA,.          /*STORE DERIVATIVE VALUE Z(DIM)*/DGT3 520
         END,.                                                          DGT3 530
   ELSE LERR ='2',.                   /*ERROR IN SPECIFIED DIMENSION */DGT3 540
   ERROR=LERR,.                                                         DGT3 550
   END,.                              /*END OF PROCEDURE DGT3        */DGT3 560
```

Purpose:

DGT3 computes a vector $Z = (z_1, \ldots, z_{DIM})$ of derivative values, when vectors $X = (x_1, \ldots, x_{DIM})$ of argument values and $Y = (y_1, y_2, \ldots, y_{DIM})$ of corresponding function values are given.

Usage:

CALL DGT3 (X, Y, Z, DIM);

X(DIM) - BINARY FLOAT [(53)]
Given vector of argument values.

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of derivative values.

DIM - BINARY FIXED
Given dimension of vectors X, Y and Z.

Remarks:

If no errors are detected in the processing of data, the data indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='1' means non-monotonic argument values; that is, for some i, $(x_i-x_{i-1})\cdot(x_{i+1}-x_i)$ is less than or equal to zero.

ERROR='2' means DIM is less than three.

Vectors Z and Y may be identically allocated, which means that the given function values are replaced by the resultant derivative values.

Method:

The resultant value z_i is calculated as the derivative of the Lagrangian interpolation polynomial passing through points i-1, i, i+1, at argument x_i.

$$z_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}$$

for i = 2, 3, ..., DIM-1, and corresponding formulas for z_1, z_{DIM}.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis. McGraw-Hill, New York-Toronto-London, 1956, pp. 64-68.

Mathematical background:

For i = 1, ..., n-2 we must find a_i, b_i, and c_i such that

$$\bar{y}_i(x) = a_i x^2 + b_i x + c_i$$

passes through (x_i, y_i), (x_{i+1}, y_{i+1}), and (x_{i+2}, y_{i+2}).

The desired derivative values z_i are given by:

$$z_i = \begin{cases} y'_1(x_1) = 2a_1x_1 + b_1 & \text{if } i = 1 \\ y'_{i-1}(x_i) = 2a_{i-1}x_i + b_{i-1} & \text{if } i = 2, \ldots, n-1 \\ y'_{n-2}(x_n) = 2a_{n-2}x_n + b_{n-2} & \text{if } i = n \end{cases}$$

An easy computation yields:

$$z_i=\begin{cases}\dfrac{y_2-y_1}{x_2-x_1}+\dfrac{y_3-y_1}{x_3-x_1}-\dfrac{y_3-y_2}{x_3-x_2} & \text{if } i=1\\[2ex] \dfrac{y_i-y_{i-1}}{x_i-x_{i-1}}+\dfrac{y_{i+1}-y_i}{x_{i+1}-x_i}-\dfrac{y_{i+1}-y_{i-1}}{x_{i+1}-x_{i-1}} & \text{if } i=2,\ldots,n-1\\[2ex] \dfrac{y_n-y_{n-1}}{x_n-x_{n-1}}+\dfrac{y_n-y_{n-2}}{x_n-x_{n-2}}-\dfrac{y_{n-1}-y_{n-2}}{x_{n-1}-x_{n-2}} & \text{if } i=n\end{cases}\qquad(1)$$

Assuming that the vectors X and Y represent a portion of a three-times-differentiable function, z_i involves a truncation error T_i where:

$$T_i=\begin{cases}\frac{1}{6}(x_1-x_2)(x_1-x_3)y'''(\xi_1) & \text{if } i=1\\[1ex] \frac{1}{6}(x_i-x_{i-1})(x_i-x_{i+1})y'''(\xi_i) & \text{if } i=2,\ldots,n-1\\[1ex] \frac{1}{6}(x_n-x_{n-2})(x_n-x_{n-1})y'''(\xi_n) & \text{if } i=n\end{cases}$$

and ξ_i is in the closed interval determined by the three argument values used in computing z_i, $i = 1,\ldots,n$.

Programming Considerations:

The given table should represent a single-valued function. Non-monotonic arguments may cause dubious derivative values. If any difference (x_i-x_{i-1}), $(x_{i+1}-x_i)$, $(x_{i+1}-x_{i-1})$ is zero, it is replaced by 10^{-30}.

● Subroutine DET3

```
DET3..
/**********************************************************************/
/*                                                                    */
/*      DIFFERENTIATE AN EQUIDISTANTLY TABLED FUNCTION USING          */
/*      LAGRANGIAN INTERPOLATION FORMULA, DEGREE 2                    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(H,Y,Z,DIM),.
   DECLARE
        (H,Y(*),Z(*),YA,YB,YC,HH)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        (DIM,I)BINARY FIXED,
        ERROR EXTERNAL CHARACTER(1),.
   IF DIM GE 3                         /*TEST SPECIFIED DIMENSION     */
   THEN DO,.
        IF H NE 0                      /*TEST SPECIFIED INCREMENT     */
        THEN DO,.
             HH   =H+H,.
             YC   =Y(1),.
             YA   =YC-Y(2),.
             YB   =Y(3)+YA+YA+YA,.     /*MODIFICATION  YB = Y(0)      */
                  DO I =2 TO DIM,.
                  YA   =YB,.           /*REPLACE YA BY Y(I-2)         */
                  YB   =YC,.           /*REPLACE YB BY Y(I-1)         */
                  YC   =Y(I),.         /*SET YC TO Y(I)               */
                  Z(I-1)=(YC-YA)/HH,.  /*SET Z(I-1) TO (Y(I)-Y(I-2)/2H*/
                  END,.
             YC   =YC-YB,.
             Z(DIM)=(YA-YB+YC          /*Z(DIM)=(Y(DIM-2)-4*Y(DIM-1)  */
                  +YC+YC)/HH,.         /**+3*Y(DIM))2*H               */
             ERROR='0',.               /*SUCCESSFUL OPERATION         */
             END,.
        ELSE ERROR='1',.               /*ERROR IN SPECIFIED INCREMENT */
        END,.
   ELSE ERROR='2',.                    /*ERROR IN SPECIFIED DIMENSION */
   END,.                               /*END OF PROCEDURE DET3        */
```

Purpose:

DET3 computes a vector Z = $(z_1, z_2, \ldots, z_{DIM})$ of derivative values, given a vector Y = $(y_1, y_2, \ldots, y_{DIM})$ of function values whose components y_i correspond to DIM equidistantly spaced argument values x_i with x_i-x_{i-1} = h for i = 2, ..., DIM.

Usage:

CALL DET3 (H, Y, Z, DIM);

H - BINARY FLOAT [(53)]
Given increment of argument values.
Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.
Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of derivative values.
DIM - BINARY FIXED
Given dimension of vector Y and Z.

Remarks:

If no errors are detected in the processing of data, the data indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='1' means DIM is less than three.
ERROR='2' means H is equal to zero.

Storage allocation for the vectors Z and Y may be identical, which means that the given function values are replaced by the resultant derivative values.

Method:

The resultant value z_i is calculated as the derivative of the Lagrangian interpolation polynomial passing through the points i-1, i, i+1 at argument x_i.

$$z_i = \frac{1}{2h}(y_{i+1} - y_{i-1}) \text{ for } i = 2, 3, \ldots, \text{DIM}-1$$

and corresponding formulas for z_1, z_{DIM}.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 82-84.

Mathematical Background:

The procedure is described under subroutine DGT3, but here we have the additional relation $x_i - x_{i-1} = h$, a constant, for $i = 2, \ldots, n$. This leads to the following expression for z_i:

$$z_i = \begin{cases} \frac{1}{2h}(-y_3 + 4y_2 - 3y_1) & \text{if } i=1 \\ \frac{1}{2h}(y_{i+1} - y_{i-1}) & \text{if } i=2,\ldots,n-1 \\ \frac{1}{2h}(3y_n - 4y_{n-1} + y_{n-2}) & \text{if } i=n \end{cases} \qquad (1)$$

Assuming that the vector Y represents the function values of a portion of a three-times-differentiable function, z_i involves a truncation error T_i where:

$$T_i = \begin{cases} \frac{h^2}{3} y'''(\xi_1),\ \xi_1 \in [x_1, x_3] & \text{if } i=1 \\ \frac{-h^2}{6} y'''(\xi_i),\ \xi_i \in [x_{i-1}, x_{i+1}] & \text{if } i=2,\ldots,n-1 \\ \frac{h^2}{3} y'''(\xi_n),\ \xi_n \in [x_{n-2}, x_n] & \text{if } i=n \end{cases}$$

In addition to these truncation errors, roundoff errors may be of considerable magnitude. Supposing that each of the ordinates y_i can be in error by $\pm \epsilon$, $\epsilon > 0$, the magnitude $|R_i|$ of the corresponding error R_i in the calculation of z_i can be as large as:

$$|R_i| = \begin{cases} \frac{4\epsilon}{|h|} & \text{if } i=1, n \\ \frac{\epsilon}{|h|} & \text{if } i=2,\ldots,n-1 \end{cases}$$

Since small truncation errors generally require small $|h|$, while small roundoff errors generally require large $|h|$, it is reasonable to choose h so that $|T_i| \approx |R_i|$.

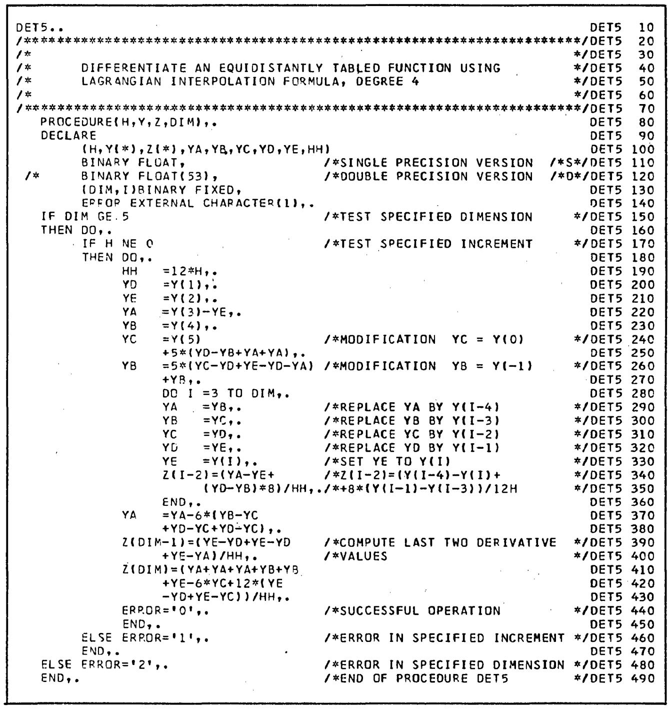
If $M = \sup y'''(\xi)$, where $\xi \in [x_1, x_n]$, and if we regard only the inner points $x_2, \ldots, x_{n-1}$, we find that

$$h_{\text{optimum}} \approx \pm 1.8 \sqrt[3]{\epsilon/M}$$

and the magnitude $|E_i|$ of the total possible error E_i in z_i is given by:

$$|E_i| \approx \begin{cases} 3.3 \sqrt[3]{\epsilon^2 M} & \text{if } i=1, n \\ 1.1 \sqrt[3]{\epsilon^2 M} & \text{if } i=2,\ldots,n-1 \end{cases}$$

- Subroutine DET5



```
DET5..
/*********************************************************************/
/*                                                                  */
/*      DIFFERENTIATE AN EQUIDISTANTLY TABLED FUNCTION USING        */
/*      LAGRANGIAN INTERPOLATION FORMULA, DEGREE 4                  */
/*                                                                  */
/*********************************************************************/
   PROCEDURE(H,Y,Z,DIM),.
   DECLARE
        (H,Y(*),Z(*),YA,YB,YC,YD,YE,HH)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        (DIM,I)BINARY FIXED,
        EPFOP EXTERNAL CHAPACTER(1),.
   IF DIM GE.5                         /*TEST SPECIFIED DIMENSION     */
   THEN DO,.
        IF H NE 0                      /*TEST SPECIFIED INCREMENT     */
        THEN DO,.
             HH   =12*H,.
             YD   =Y(1),.
             YE   =Y(2),.
             YA   =Y(3)-YE,.
             YB   =Y(4),.
             YC   =Y(5)                /*MODIFICATION  YC = Y(0)      */
                  +5*(YD-YB+YA+YA),.
             YB   =5*(YC-YD+YE-YD-YA)  /*MODIFICATION  YB = Y(-1)     */
                  +YB,.
                  DO I =3 TO DIM,.
                  YA   =YB,.           /*REPLACE YA BY Y(I-4)         */
                  YB   =YC,.           /*REPLACE YB BY Y(I-3)         */
                  YC   =YD,.           /*REPLACE YC BY Y(I-2)         */
                  YD   =YE,.           /*REPLACE YD BY Y(I-1)         */
                  YE   =Y(I),.         /*SET YE TO Y(I)               */
                  Z(I-2)=(YA-YE+       /*Z(I-2)=(Y(I-4)-Y(I)+         */
                       (YD-YB)*8)/HH,./*+8*(Y(I-1)-Y(I-3))/12H        */
                  END,.
             YA   =YA-6*(YB-YC
                  +YD-YC+YD-YC),.
             Z(DIM-1)=(YE-YD+YE-YD     /*COMPUTE LAST TWO DERIVATIVE  */
                  +YE-YA)/HH,.         /*VALUES                       */
             Z(DIM)=(YA+YA+YA+YB+YB
                  +YE-6*YC+12*(YE
                  -YD+YE-YC))/HH,.
             ERROR='0',.               /*SUCCESSFUL OPERATION         */
             END,.
        ELSE ERROR='1',.               /*ERROR IN SPECIFIED INCREMENT */
        END,.
   ELSE ERROR='2',.                    /*ERROR IN SPECIFIED DIMENSION */
   END,.                               /*END OF PROCEDURE DET5        */
```

Purpose:

DET5 computes a vector $Z = (z_1, z_2, \dots, z_{DIM})$ of derivative values, given $Y = (y_1, y_2, \dots, y_{DIM})$ of function values whose components correspond to DIM equidistantly spaced argument values x_i, with $x_i - x_{i-1} = h$.

Usage:

CALL DET5 (H, Y, Z, DIM);

H - BINARY FLOAT [(53)]
Given increment for argument values.

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of derivative values.

DIM - BINARY FIXED
Given dimension of vectors Y and Z.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR='1' means H is equal to zero.
ERROR='2' means DIM is less than five.

Storage allocation for vectors Z and Y may be identical, which means that the given function values are replaced by the resultant derivative values.

Method:

The resultant value z_i is calculated as the derivative of the Lagrangian interpolation polynomial passing through the points i-2, i-1, i, i+1, i+2 at argument x_i.

$$z_i = \frac{1}{2h}(y_{i+1} - y_{i-1}) \quad \text{for } i = 3, 4, \dots, DIM-2$$

and corresponding formulas for z_1, z_2, z_{DIM-1}, z_{DIM}

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 82-84.

Mathematical Background:

For i = 1,..., n-4 we must find a_i, b_i, c_i, d_i, and e_i such that

$$\bar{y}_i(x) = a_i x^4 + b_i x^3 + c_i x^2 + d_i x + e_i$$

passes through (x_{i+k}, y_{i+k}) for $k = 0, \dots, 4$.

The desired derivative values z_i are given by:

$$z_i = \begin{cases} \bar{y}'_1(x_i) = 4a_1x_i^3 + 3b_1x_i^2 + 2c_1x_i + d_1 & \text{if } i=1,\ 2 \\ \bar{y}'_{i-2}(x_i) = 4a_{i-2}x_i^3 + 3b_{i-2}x_i^2 + 2c_{i-2}x_i + d_{i-2} & \text{if } i=3,\dots,n-2 \\ \bar{y}'_{n-4}(x_i) = 4a_{n-4}x_i^3 + 3b_{n-4}x_i^2 + 2c_{n-4}x_i + d_{n-4} & \text{if } i=n-1,\ n \end{cases}$$

Using the fact that $x_i - x_{i-1} = h$, a constant, for $i=2,\ldots,n$, we get:

$$z_i = \begin{cases} \frac{1}{12h}(-25y_1 + 48y_2 - 36y_3 + 16y_4 - 3y_5) & \text{if } i=1 \\ \frac{1}{12h}(-3y_1 - 10y_2 + 18y_3 - 6y_4 + y_5) & \text{if } i=2 \\ \frac{1}{12h}(y_{i-2} - 8y_{i-1} + 8y_{i+1} - y_{i+2}) & \text{if } i=3,\ldots,n-2 \\ \frac{1}{12h}(-y_{n-4} + 6y_{n-3} - 18y_{n-2} + 10_{n-1} + 3y_n) & \text{if } i=n-1 \\ \frac{1}{12h}(3y_{n-4} - 16y_{n-3} + 36y_{n-2} - 48y_{n-1} + 25y_n) & \text{if } i=n \end{cases} \qquad (1)$$

Assuming that the vector Y represents the function values of a portion of a five-times-differentiable function, z_i involves a truncation error T_i where:

$$T_i = \begin{cases} \frac{h^4}{5} y^V(\xi_1), & \xi_1 \epsilon [x_1, x_5] \quad \text{if } i=1 \\ -\frac{h^4}{20} y^V(\xi_2), & \xi_2 \epsilon [x_1, x_5] \quad \text{if } i=2 \\ \frac{h^4}{30} y^V(\xi_i), & \xi_i \epsilon [x_{i-2}, x_{i+2}] \quad \text{if } i=3,\ldots,n-2 \\ -\frac{h^4}{20} y^V(\xi_{n-1}), & \xi_{n-1} \epsilon [x_{n-4}, x_n] \quad \text{if } i=n-1 \\ \frac{h^4}{5} y^V(\xi_n), & \xi_n \epsilon [x_{n-4}, x_n] \quad \text{if } i=n \end{cases}$$

In addition to the truncation errors, roundoff errors may be of considerable magnitude. Supposing that the ordinates y_i can be in error by $\pm \epsilon$, $\epsilon > 0$, the magnitude $|R_i|$ of the corresponding error R_i in the computation of z_i can be as large as:

$$|R_i| = \begin{cases} \frac{32\epsilon}{3|h|} & \text{if } i=1, n \\ \frac{19\epsilon}{6|h|} & \text{if } i=2, n-1 \\ \frac{3\epsilon}{2|h|} & \text{if } i=3, \ldots, n-2 \end{cases}$$

Since small truncation errors generally require small $|h|$, while small roundoff errors generally require large $|h|$, it is reasonable to choose h so that $|T_i| \approx |R_i|$.

$$\text{If } M = \sup_{\xi \,\epsilon\, [x_1, x_n]} y^V(\xi)$$

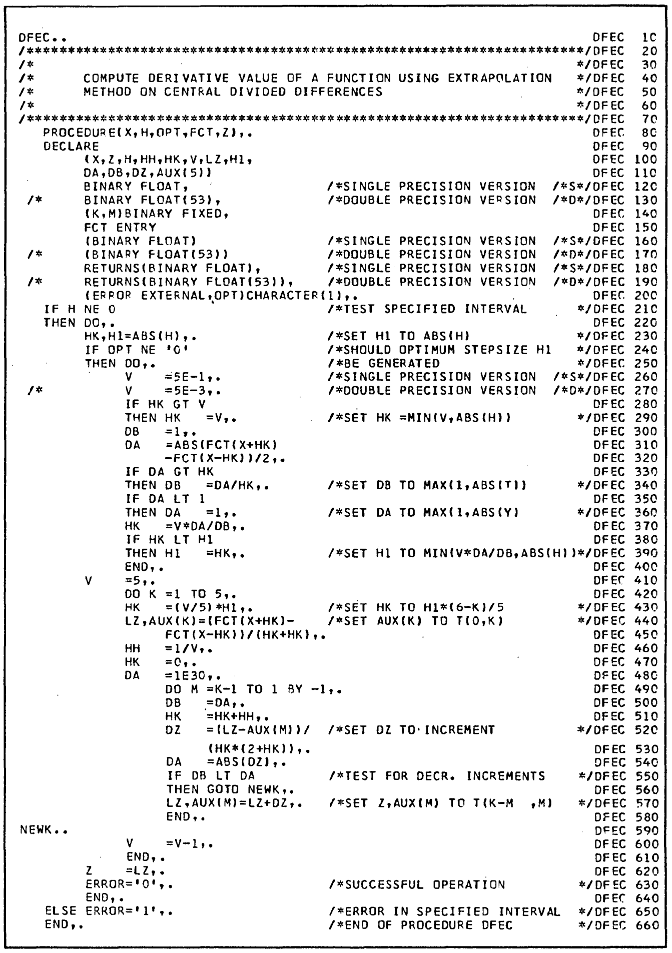
and if we regard only the inner points $x_3, \ldots, x_{n-2}$, we find that

$$h_{\text{optimum}} \approx 2.1 \sqrt[5]{\epsilon/M}$$

and the magnitude $|E_i|$ of the total possible error E_i in z_i is given by:

$$|E_i| \approx \begin{cases} 9 \sqrt[5]{\epsilon^4 M} & \text{if } i = 1, n \\ 2.5 \sqrt[5]{\epsilon^4 M} & \text{if } i = 2, n-1 \\ 1.4 \sqrt[5]{\epsilon^4 M} & \text{if } i = 3, \ldots, n-2 \end{cases}$$

- Subroutine DFEC



```
DFEC..                                                                DFEC  10
/********************************************************************/DFEC  20
/*                                                                  */DFEC  30
/*      COMPUTE DERIVATIVE VALUE OF A FUNCTION USING EXTRAPOLATION  */DFEC  40
/*      METHOD ON CENTRAL DIVIDED DIFFERENCES                       */DFEC  50
/*                                                                  */DFEC  60
/********************************************************************/DFEC  70
   PROCEDURE(X,H,OPT,FCT,Z),.                                         DFEC  80
   DECLARE                                                            DFEC  90
        (X,Z,H,HH,HK,V,LZ,H1,                                         DFEC 100
        DA,DB,DZ,AUX(5))                                              DFEC 110
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION /*S*/DFEC 120
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION /*D*/DFEC 130
        (K,M)BINARY FIXED,                                            DFEC 140
        FCT ENTRY                                                     DFEC 150
        (BINARY FLOAT)                /*SINGLE PRECISION VERSION /*S*/DFEC 160
 /*     (BINARY FLOAT(53))            /*DOUBLE PRECISION VERSION /*D*/DFEC 170
        RETURNS(BINARY FLOAT),        /*SINGLE PRECISION VERSION /*S*/DFEC 180
 /*     RETURNS(BINARY FLOAT(53)),    /*DOUBLE PRECISION VERSION /*D*/DFEC 190
        (ERROR EXTERNAL,OPT)CHARACTER(1),.                            DFEC 200
   IF H NE 0                          /*TEST SPECIFIED INTERVAL     */DFEC 210
   THEN DO,.                                                          DFEC 220
        HK,H1=ABS(H),.                /*SET H1 TO ABS(H)            */DFEC 230
        IF OPT NE '0'                 /*SHOULD OPTIMUM STEPSIZE H1  */DFEC 240
        THEN DO,.                     /*BE GENERATED                */DFEC 250
             V    =5E-1,.             /*SINGLE PRECISION VERSION /*S*/DFEC 260
 /*          V    =5E-3,.             /*DOUBLE PRECISION VERSION /*D*/DFEC 270
             IF HK GT V                                               DFEC 280
             THEN HK   =V,.           /*SET HK =MIN(V,ABS(H))       */DFEC 290
             DB   =1,.                                                DFEC 300
             DA   =ABS(FCT(X+HK)                                      DFEC 310
                  -FCT(X-HK))/2,.                                     DFEC 320
             IF DA GT HK                                              DFEC 330
             THEN DB   =DA/HK,.       /*SET DB TO MAX(1,ABS(T))     */DFEC 340
             IF DA LT 1                                               DFEC 350
             THEN DA   =1,.           /*SET DA TO MAX(1,ABS(Y)      */DFEC 360
             HK   =V*DA/DB,.                                          DFEC 370
             IF HK LT H1                                              DFEC 380
             THEN H1   =HK,.          /*SET H1 TO MIN(V*DA/DB,ABS(H))*/DFEC 390
             END,.                                                    DFEC 400
        V    =5,.                                                     DFEC 410
             DO K =1 TO 5,.                                           DFEC 420
             HK   =(V/5)*H1,.         /*SET HK TO H1*(6-K)/5        */DFEC 430
             LZ,AUX(K)=(FCT(X+HK)-    /*SET AUX(K) TO T(0,K)        */DFEC 440
                  FCT(X-HK))/(HK+HK),.                                DFEC 450
             HH   =1/V,.                                              DFEC 460
             HK   =0,.                                                DFEC 470
             DA   =1E30,.                                             DFEC 480
                  DO M =K-1 TO 1 BY -1,.                              DFEC 490
                  DB   =DA,.                                          DFEC 500
                  HK   =HK+HH,.                                       DFEC 510
                  DZ   =(LZ-AUX(M))/  /*SET DZ TO INCREMENT         */DFEC 520
                       (HK*(2+HK)),.                                  DFEC 530
                  DA   =ABS(DZ),.                                     DFEC 540
                  IF DB LT DA         /*TEST FOR DECR. INCREMENTS   */DFEC 550
                  THEN GOTO NEWK,.                                    DFEC 560
                  LZ,AUX(M)=LZ+DZ,.   /*SET Z,AUX(M) TO T(K-M  ,M)  */DFEC 570
                  END,.                                               DFEC 580
NEWK..                                                                DFEC 590
             V    =V-1,.                                              DFEC 600
             END,.                                                    DFEC 610
        Z    =LZ,.                                                    DFEC 620
        ERROR='0',.                   /*SUCCESSFUL OPERATION        */DFEC 630
        END,.                                                         DFEC 640
   ELSE ERROR='1',.                   /*ERROR IN SPECIFIED INTERVAL */DFEC 650
   END,.                              /*END OF PROCEDURE DFEC       */DFEC 660
```

Purpose:

Given the argument X and the function FCT(X), defined in the closed interval [X-|H| , X+ |H|]. DFEC computes an approximation Z to the derivative of the function FCT(X).

Usage:

CALL DFEC (X, H, OPT, FCT, Z);

X - BINARY FLOAT [(53)]
Given argument value.
H - BINARY FLOAT [(53)]
Given radius of closed interval about X.
OPT - CHARACTER (1)
Given option for calculation of the stepsize.
FCT - ENTRY
Given procedure for calculating of function values, which must be supplied by the user.
Usage:
FCT(T)
FCT(T) - BINARY FLOAT [(53)]
Resultant function value.
T - BINARY FLOAT [(53)]
Given argument value.

Z - BINARY FLOAT [(53)]
Resultant approximation to $\frac{d}{dx}$ FCT(X).

Remarks:

OPT = '0' means maximum stepsize is set equal to H; otherwise, it will be calculated within procedure DFEC (for details see "Mathematical Background").

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR = '1' means given H is equal to zero.

Method:

The approximation Z of the derivative is obtained by applying Richardson's and Romberg's extrapolation method to successively computed central divided differences, using function values in the closed interval [X-|H|, X+|H|].

For reference see:

S. Fillipi and H. Engels, "Altes und Neues zur numerischen Differentiation", Elektronische Datenverarbeitung, iss. 2 (1966), pp. 57 - 65.

Mathematical Background:

Suppose, first, that y = y(t) is analytic at x; that is, y has a Taylor series expansion about the point x with radius of convergence R > 0. Let h be such that 0 < | h | < R. For each positive integer n a step size h_1 with $0 < h_1 \leq |h|$ is computed as described below, and a sequence h_k of increments is generated, where

$$h_k = \frac{n-k+1}{n} h_1 \qquad \text{for } k=2,\dots,n$$

From the sequence $(x-h_k,\ x+h_k)$ of point pairs (k=1,..., n), the sequence of central divided differences

$$T_{0,k} = \frac{y(x+h_k) - y(x-h_k)}{2h_k} \quad \text{for } k=1,\dots,n \qquad (1)$$

is computed, which forms the first column of the triangular Romberg scheme. The central divided differences $T_{0,k}$ represent the slopes of the secants s_k in Figure 2.

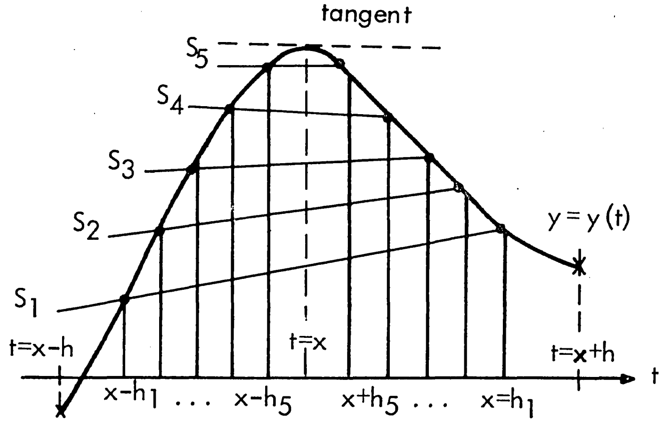


Figure 2. A sequence of secants for a given function y=y(t) and a given argument t=x for the case n=5, h > 0

From the Taylor series expansions of $y(x+h_k)$ and $y(x-h_k)$ it follows that

$$T_{0,k} = y'(x) + \frac{h_k^2}{3!} y'''(x) + \frac{h_k^4}{5!} y^{v}(x) + \dots$$

for k=1,..., n

so that, as an approximation to $y'(x)$, $T_{0,k}$ involves a truncation error of order h_k^2.

Knowing the two divided differences $T_{0,k}$ and $T_{0,k+1}$, we are able to generate the extrapolated value

$$T_{1,k} = T_{0,k+1} + \frac{T_{0,k+1} - T_{0,k}}{a_{1,k}^{-1}}$$

where (2)

$$a_{1,k} = (1 + \frac{1}{n-k})^2$$

$T_{1,k}$ is a better approximation to $y'(x)$ since

$$T_{1,k} = y'(x) - \frac{1}{5!} \frac{1}{a_{1,k}} h_k^4 y^{v}(x) - \frac{1}{7!} \frac{1}{a_{1,k}} (1 + \frac{1}{a_{1,k}}) h_k^6 y^{vii}(x) - \dots$$

which involves a truncation error of order h_k^4.

If we also know $T_{0,k+2}$, we can generate $T_{1,k+1}$ using equation (2), and further, we can compute the extrapolated value

$$T_{2,k} = T_{1,k+1} + \frac{T_{1,k+1} - T_{1,k}}{a_{2,k}^{-1}}$$

where

$$a_{2,k} = (1 + \frac{2}{n-(k+1)})^2$$

which involves a truncation error of order h_k^6.

Generally, the order of the truncation error is increased by 2 with each new extrapolation step; in particular, $T_{i,j}$ will involve a truncation error of order

$$h_j^{2i+2}, \ i = 0, \dots, n-1, \ j = 1, \dots, n.$$

Figure 3 shows the arrangement of the T values in the triangular Romberg scheme. The T values are computed following the upward diagonals, using the general formula:

$$T_{m,k-m} = T_{m-1,k-m+1} + \frac{T_{m-1,k-m+1} - T_{m-1,k-m}}{\frac{m}{n-(k+1)} \left(2 + \frac{m}{n-(k+1)}\right)} \qquad (4)$$

for m = 1,..., k-1 for fixed k, k=2, ..., n

Truncation error		$O(h_k^2)$	$O(h_k^4)$	$O(h_k^6)$	$O(h_k^8)$	$O(h^{10}_k)$
Stepsize	m k	0	1	2	3	4
h_1	1	$T_{0,1}$	$T_{1,1}$	$T_{2,1}$	$T_{3,1}$	$T_{4,1}$
$h_2=0.8h_1$	2	$T_{0,2}$	$T_{1,2}$	$T_{2,2}$	$T_{3,2}$	
$h_2=0.6h_1$	3	$T_{0,3}$	$T_{1,3}$	$T_{2,3}$		
$h_4=0.4h_1$	4	$T_{0,4}$	$T_{1,4}$			
$h_5=0.2h_1$	5	$T_{0,5}$				

Figure 3. The triangular Romberg scheme of T-values for the case n=5

Numerical experience shows that the accuracy of the results depends heavily on roundoff errors in the central divided differences $T_{0,k}$. Therefore, the choice of the absolutely smallest step size, h_n, is based on the following considerations.
Let:

$$v = \begin{cases} 1 \text{ in single-precision computation} \\ 3 \text{ in double-precision computation} \end{cases}$$

$$h_0 = \min\,(n \cdot 10^{-v}, |h|)$$

Set:

$$Y = \frac{1}{2}(y(x+h_0) + y(x-h_0))$$

and

$$T = \frac{1}{2}(y(x+h_0) - y(x-h_0))/h_0$$

Y and T are approximations to y(x) and y'(x), respectively.

Assuming that the errors in the function values y(t) for $t \epsilon [x-|h|, x+|h|]$ are bounded by

$$\epsilon = \begin{cases} Y \cdot 10^{-D} & \text{if } |Y| \geq 1 \\ 10^{-D} & \text{if } |Y| \leq 1 \end{cases}$$

formula (1) shows that the roundoff error in the computation of $T_{0,n}$ is bounded by

$$R(T_{0,n}) = \frac{\epsilon}{h_n} = \begin{cases} \dfrac{Y \cdot 10^{-D}}{h_n} & \text{if } |Y| > 1 \\ \dfrac{10^{-D}}{h_n} & \text{if } |Y| \leq 1 \end{cases}$$

where D is the number of significant digits in the floating-point representation of numbers. Suppose, also, that we are willing to tolerate a roundoff error

$$R'(T_{0,n}) = \begin{cases} T \cdot 10^{-D+v} & \text{if } |T| > 1 \\ 10^{-D+v} & \text{if } |T| \leq 1 \end{cases}$$

Then we must have $R(T_{0,n}) \leq R'(T_{0,n})$, which is satisfied when

$$h_n = \frac{\max(1, |Y|)}{\max(1, |T|)} \cdot 10^{-v} \qquad (5)$$

Finally we set

$$h_1 = \min(n \cdot h_n, |h|) \qquad (6)$$

guaranteeing that the evaluation of the function y = y(t) is restricted to the closed interval $[x-|h|, x+|h|]$.

Programming Considerations:

Numerical experience shows that, because of increasing roundoff errors, it is generally fruitless to perform more than five extrapolations. Thus, the subroutine uses n = 5, and it is therefore necessary only that y = y(t) be eleven-times differentiable, rather than analytic. It is easy to see that in the case n = 5, y = y(t) must be evaluated at twelve points in the closed interval $[x-|h|, x+|h|]$.

As previously explained, the computation of the T values is performed along the upward diagonals of the triangular Romberg scheme. Therefore, only a one-dimensional internal storage vector, named AUX, with five storage locations is necessary. Figure 4 shows the storage administration and the sequence of computations (numbers in parentheses).

AUX(1)	$T_{0,5}^{(11)}$				
AUX(2)	$T_{0,4}^{(7)}$	$T_{1,4}^{(12)}$			
AUX(3)	$T_{0,3}^{(4)}$	$T_{1,3}^{(8)}$	$T_{2,3}^{(13)}$		
AUX(4)	$T_{0,2}^{(2)}$	$T_{1,2}^{(5)}$	$T_{2,2}^{(9)}$	$T_{3,2}^{(14)}$	
AUX(5)	$T_{0,1}^{(1)}$	$T_{1,1}^{(3)}$	$T_{2,1}^{(6)}$	$T_{3,1}^{(10)}$	$T_{4,1}^{(15)}$

Figure 4. Storage administration and order of computation

Each extrapolation loop, the computation of the elements on an upward diagonal, is terminated as soon as the absolute values of the differences between adjacent diagonal elements stop decreasing, showing the influence of roundoff errors. The computed T value that differs least in absolute value from its immediately preceding diagonal neighbor is the desired value Z.

Subroutine DFEO

```
DFEO..
/**********************************************************************/
/*                                                                    */
/*      COMPUTE DERIVATIVE VALUE OF A FUNCTION USING EXTRAPOLATION    */
/*      METHOD CN ONE-SIDED DIVIDED DIFFERENCES                       */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,H,OPT,FCT,Z),.
   DECLARE
        (X,Z,H,HK,HH,V,Y,H1,
        DA,DB,DZ,AUX(1C))
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION   /*S*/
 /*     BINAFY FLOAT(53),             /*DOUBLE PRECISION VERSION   /*D*/
        (K,M)BINARY FIXED,
        FCT ENTRY
        (BINAPY FLOAT)                /*SINGLE PRECISION VERSION   /*S*/
 /*     (BINARY FLOAT(53))            /*DOUBLE PRECISION VERSION   /*D*/
        RETURNS(BINARY FLOAT),        /*SINGLE PRECISION VERSION   /*S*/
 /*     FETURNS(BINARY FLOAT(53)),    /*DOUBLE PRECISION VERSION   /*D*/
        (ERROR EXTERNAL,OPT)CHARACTER(1),.
   IF H NE O                          /*TEST SPECIFIED INTERVAL       */
   THEN DO,.
        H1   =H,.
        Y    =FCT(X),.
        IF OPT NE 'C'                 /*SHOULD OPTIMUM STEPSIZE H1    */
        THEN DO,.                     /*BE GENERATED                  */
             V    =5E-1,.             /*SINGLE PRECISION VERSION   /*S*/
 /*          V    =5E-3,.             /*DOUBLE PRECISION VERSION   /*D*/
             IF H1 LT C
             THEN V    =-V,.
             IF ABS(V) GT ABS(H1)
             THEN HH   =H1,.          /*SET HH=SIGN(H)*MIN(V,ABS(H)) */
             ELSE HH   =V,.
             DB   =ABS((FCT(X+HH)
                  -Y)/HH),.
             IF DB LT 1
             THEN DB   =1,.           /*SET DB TO MAX(1,ABS(T))       */
             HK   =(V+V)/DB,.
             IF ABS(Y) GT 1
             THEN HK   =HK*ABS(Y),.   /*SET HK=2*V*MAX(1,ABS(Y))/DB   */
             IF ABS(HK) LT ABS(H1)
             THEN H1   =HK,.          /*SET H1=SIGN(H)*MIN(HK,ABS(H))*/
             END,.
        V    =10,.
             DO K =1 TO 10,.
             HK   =(V/10)*H1,.        /*SET HK TO H1*(11-K)/10        */
             Z,AUX(K)=(FCT(X+HK)-Y)   /*SET AUX(K) TO T(0,K)          */
                  /HK,.
             HH   =1/V,.
             HK   =C,.
             DA   =1E3C,.
                  DO M =K-1 TO 1 BY -1,.
                  HK   =HK+HH,.
                  DZ   =(Z-AUX(M))
                       /HK,.          /*SET DZ TO INCREMENT           */
                  DB   =DA,.
                  DA   =ABS(DZ),.
                  IF DB LT DA         /*TEST FOR DECREASING INCREMENT*/
                  THEN GOTO NEWK,.
                  Z,AUX(M)=Z+DZ,.     /*SET Z,AUX(M) TO T(K-M,M)      */
                  END,.
NEWK..
             V    =V-1,.
             END,.
        EFFOR='C',.                   /*SUCCESSFUL OPERATION          */
        END,.
   ELSE ERROR='1',.                   /*ERROR IN SPECIFIED INTERVAL   */
   END,.                              /*END OF PROCEDURE DFEO         */
```

Purpose:

Given argument X and function FCT(X), defined in the one-sided interval [X,X+H], DFEO computes an approximation Z to the derivative.

Usage:

CALL DFEO (X, H, OPT, FCT, Z);

X - BINARY FLOAT [(53)]
Given argument value.

H - BINARY FLOAT [(53)]
Given length of interval.

OPT - CHARACTER (1)
Given option for calculation of the stepsize.

FCT - ENTRY
Given procedure for calculation of function values, which must be supplied by the user.

Usage:
FCT(T)

FCT(T) - BINARY FLOAT [(53)]
Resultant function value.

T - BINARY FLOAT [(53)]
Given argument value.

Z - BINARY FLOAT [(53)]
Resultant approximation to $\frac{d}{dx}$ FCT(X).

Remarks:

OPT = '0' means maximum stepsize is set equal to H; otherwise, it will be calculated within procedure DFEO (for details see "Mathematical Background").

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR = '1' means H is equal to zero.

Method:

The approximation Z of the derivative is obtained by applying Richardson's and Romberg's extrapolation method to successively computed one-sided divided differences, using function values in the closed interval [X, X+H].

For reference see:

S. Fillipi and Engels, "Altes und Neues zur numerischen Differentiation", Elektronische Datenverarbeitung, iss. 2 (1966), pp. 57-65.

Mathematical Background:

Suppose, first, that y=y(t) is analytic at x; that is, y has a Taylor series expansion about the point x with radius of convergence $R > 0$. Let h be such that $0 < |h| < R$. For each positive integer n, a stepsize h_1 with $0 < |h_1| \leq |h|$ is computed as described below, and a sequence h_k of increments is generated, where

$$h_k = \frac{n-k+1}{n} h_1$$

for k = 2, ..., n.

From the sequence $(x, x+h_k)$ of point pairs (k = 1,..., n), the sequence of one-sided divided differences

$$T_{0,k} = \frac{y(x+h_k) - y(x)}{h_k} \quad \text{for } k = 1, \ldots, n \tag{1}$$

is computed, which forms the first column of the triangular Romberg scheme. These one-sided divided differences $T_{0,k}$ represent the slopes of the secants s_k in Figure 5 in the case $h > 0$.

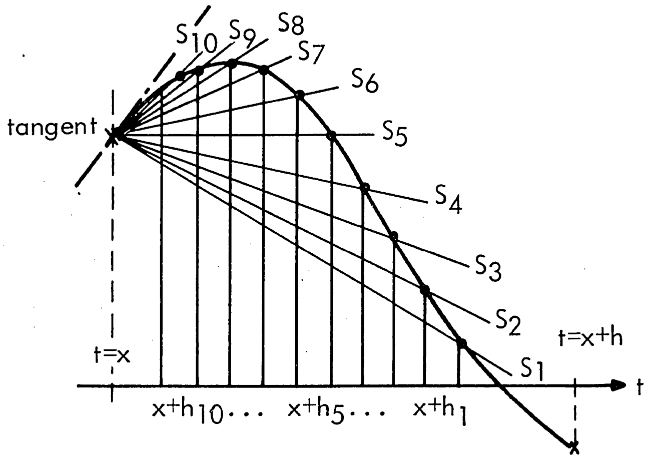


Figure 5. A sequence of secants for a given function $y = y(t)$ and a given argument $t = x$ for the case $n = 10, h > 0$

From the Taylor series expansion of $y(x+h_k)$ it follows that

$$T_{0,k} = y'(x) + \frac{h_k}{2} y''(x) + \frac{h_k^2}{3!} y'''(x) + \ldots \qquad \text{for } k = 1, \ldots, n$$

so that, as an approximation to $y'(x)$, $T_{0,k}$ involves a truncation error of order h_k. Knowing the two divided differences $T_{0,k}$ and $T_{0,k+1}$, we are able to generate the extrapolated value

$$T_{1,k} = T_{0,k+1} + \frac{T_{0,k+1} - T_{0,k}}{a_{1,k}^{-1}} \qquad (2)$$

where

$$a_{1,k} = \left(1 + \frac{1}{n-k}\right)$$

$T_{1,k}$ is a better approximation to $y'(x)$ since

$$T_{1,k} = y'(x) - \frac{1}{3!} \frac{1}{a_{i,k}^{-1}} h_k^2 y'''(x)$$

$$- \frac{1}{4!} \frac{1}{a_{i,k}} \left(1 + \frac{1}{a_{i,k}}\right) h_k^3 y^{iv}(x) - \ldots$$

which involves a truncation error of order h_k^2.

If we also know $T_{0,k+2}$, we can generate $T_{1,k+1}$ using equation (2), and further, we can compute the extrapolated value

$$T_{2,k} = T_{1,k+1} + \frac{T_{1,k+1} - T_{1,k}}{a_{2,k}^{-1}}$$

where

$$a_{2,k} = \left(1 + \frac{2}{n-(k+1)}\right)$$

which involves a truncation error of order h_k^3.

Generally, the order of the truncation error is increased by 1 with each new extrapolation step; in particular, $T_{i,j}$ will involve a truncation error of order

$$h_j^{i+1}, \ i = 0, \ldots, n-1, \ j = 1, \ldots, n.$$

Figure 6 shows the arrangement of the T values in the triangular Romberg scheme. The T values are computed following the upward diagonals, using the general formula

$$T_{m,k-m} = T_{m-1,k-m+1} + \frac{T_{m-1,k-m+1} - T_{m-1,k-m}}{\frac{m}{n-k+1}} \qquad (3)$$

for $m = 1, \ldots, k-1$ for fixed k, $k = 2, \ldots, n$

Truncation error		$O(h_k)$	$O(h_k^2)$	$O(h_k^3)$	$O(h_k^4)$	$O(h_k^5)$	$O(h_k^6)$	$O(h_k^7)$	$O(h_k^8)$	$O(h_k^9)$	$O(h_k^{10})$
Stepsize	k \ m	0	1	2	3	4	5	6	7	8	9
h_1	1	$T_{0,1}$	$T_{1,1}$	$T_{2,1}$	$T_{3,1}$	$T_{4,1}$	$T_{5,1}$	$T_{6,1}$	$T_{7,1}$	$T_{8,1}$	$T_{9,1}$
$h_2=0.9h_1$	2	$T_{0,2}$	$T_{1,2}$	$T_{2,2}$	$T_{3,2}$	$T_{4,2}$	$T_{5,2}$	$T_{6,2}$	$T_{7,2}$	$T_{8,2}$	
$h_3=0.8h_1$	3	$T_{0,3}$	$T_{1,3}$	$T_{2,3}$	$T_{3,3}$	$T_{4,3}$	$T_{5,3}$	$T_{6,3}$	$T_{7,3}$		
$h_4=0.7h_1$	4	$T_{0,4}$	$T_{1,4}$	$T_{2,4}$	$T_{3,4}$	$T_{4,4}$	$T_{5,4}$	$T_{6,4}$			
$h_5=0.6h_1$	5	$T_{0,5}$	$T_{1,5}$	$T_{2,5}$	$T_{3,5}$	$T_{4,5}$	$T_{5,5}$				
$h_6=0.5h_1$	6	$T_{0,6}$	$T_{1,6}$	$T_{2,6}$	$T_{3,6}$	$T_{4,6}$					
$h_7=0.4h_1$	7	$T_{0,7}$	$T_{1,7}$	$T_{2,7}$	$T_{3,7}$						
$h_8=0.3h_1$	8	$T_{0,8}$	$T_{1,8}$	$T_{2,8}$							
$h_9=0.2h_1$	9	$T_{0,9}$	$T_{1,9}$								
$h_{10}=0.1h_1$	10	$T_{0,10}$									

Figure 6. The triangular Romberg scheme of T values for the case $n = 10$

Numerical experience shows that the accuracy of the results depends heavily on roundoff errors in

the one-sided divided differences $T_{0,k}$. Therefore, the choice of the absolutely smallest step size, h_n, is based on the following considerations.
Let:

$$v = \begin{cases} 1 \text{ in single-precision computation} \\ 3 \text{ in double-precision computation} \end{cases}$$

Set:

$$h_0 = \operatorname{sgn}(h) \cdot \min\left(\frac{n}{2} \cdot 10^{-v}, |h|\right)$$

$$T = (y(x+h_0) - y(x))/h_0$$

T is an approximation to $y'(x)$.

Assuming that the errors in the function values $y(t)$ for $t \epsilon\ [x, x+h]$ are bounded by

$$\epsilon = \begin{cases} |y(x)| \cdot 10^{-D} & \text{if } |y(x)| > 1 \\ 10^{-D} & \text{if } |y(x)| \le 1 \end{cases}$$

equation (1) shows that the roundoff error in the computation of $T_{0,n}$ is bounded by

$$R(T_{0,n}) = \frac{2\epsilon}{|h_n|} = \begin{cases} \dfrac{2|y(x)|\ 10^{-D}}{|h_n|} & \text{if } |y(x)| > 1 \\ 2\dfrac{10^{-D}}{|h_n|} & \text{if } |y(x)| \le 1 \end{cases}$$

where D is the number of significant digits in the floating-point representation of numbers. If we are also willing to tolerate a roundoff error

$$R'(T_{0,n}) = \begin{cases} 2T \cdot 10^{-D+v} & \text{if } |T| > 1 \\ 2\text{-}10^{-D+v} & \text{if } |T| \le 1 \end{cases}$$

we must have $R(T_{0,n}) \le R'(T_{0,n})$, which is satisfied when

$$h_n = \frac{\max(1, |y(x)|)}{\max(1, |T|)} \cdot 10^{-v} \qquad (4)$$

Finally, we set

$$h_1 - \operatorname{sgn}(h) \cdot \min(n \cdot |h_n|, |h|) \qquad (5)$$

guaranteeing that the evaluation of the function $y = y(t)$ is restricted to the closed interval $[x, x+h]$.

Programming Considerations:

Numerical experience shows that, because of increasing roundoff errors, it is generally fruitless to perform more than ten extrapolations. Thus, the subroutine uses n = 10, and it is therefore necessary only that $y = y(t)$ be eleven-times differentiable, rather than analytic. It is easy to see that in the case n = 10, $y = y(t)$ must be evaluated at twelve points in the closed interval $[x, x+h]$.

As previously explained, the computation of the T values is performed along the upward diagonals of the triangular Romberg scheme. Therefore, only a one-dimensional internal storage vector, named AUX, with ten storage locations is necessary. Figure 7 shows the storage administration and the sequence of computations (numbers in parentheses).

AUX(1)	$T_{0,10}$(46)						
AUX(2)	$T_{0,9}$(37)	$T_{1,9}$(47)					
AUX(3)	$T_{0,8}$(29)	$T_{1,8}$(38)	$T_{2,8}$(48)				
AUX(4)	$T_{0,7}$(22)	$T_{1,7}$(30)	$T_{2,7}$(39)	. . .			
AUX(5)	$T_{0,6}$(16)	$T_{1,6}$(23)	$T_{2,6}$(31)	. . .			
AUX(6)	$T_{0,5}$(11)	$T_{1,5}$(17)	$T_{2,5}$(24)	. . .			
AUX(7)	$T_{0,4}$(7)	$T_{1,4}$(12)	$T_{2,4}$(18)	. . .			
AUX(8)	$T_{0,3}$(4)	$T_{1,3}$(8)	$T_{2,3}$(13)	. . .	$T_{7,3}$(53)		
AUX(9)	$T_{0,2}$(2)	$T_{1,2}$(5)	$T_{2,2}$(9)	. . .	$T_{7,2}$(44)	$T_{8,2}$(54)	
AUX(10)	$T_{0,1}$(1)	$T_{1,1}$(3)	$T_{2,1}$(6)	. . .	$T_{7,1}$(36)	$T_{8,1}$(45)	$T_{9,1}$(55)

Figure 7. Storage administration and sequence of calculations

Each extrapolation loop, the computation of the elements on an upward diagonal, is terminated as soon as the absolute values of the differences between adjacent diagonal elements stop decreasing, showing the influence of roundoff errors. The computed T value that differs least in absolute value from its immediately preceding diagonal neighbor is the desired value Z.

Subroutine ALIM/ALIE

```
ALIM..
/*************************************************************. ********/
/*                                                                     */
/*      AITKEN SCHEME FOR INTERPOLATION OF FUNCTION VALUE              */
/*      FROM GIVEN MONOTONIC TABLE                                     */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (X,Y,DIM,ORDER,EPS,XVAL,YVAL),.
   DECLARE
         (DIM,I,J,K,N,II,JL,JR,JJL,JJR,DIMS,ORDER)
         BINARY FIXED,
         (X(*),Y(*),ARG(MIN(DIM,ORDER)),VAL(MIN(DIM,ORDER)),XVAL,
         YVAL,XST,DX,EPS,XS,Z1,Z2,D,DD,VALI,VALI1,A,DIST,DIST1,
         H,DELT1,DELT2,FACT,ARGI)
         BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*      BINARY FLOAT (53),            /*DOUBLE PRECISION VERSION  /*D*/
         (ERROR EXTERNAL,SW)
         CHARACTER (1),.
   SW    ='M',.                        /*MONOTONIC ARGUMENTS          */
   J     =1,.
   D     =1E75,.
         DO I = 1 TO DIM,.             /*COMPUTE STARTING SUBSCRIPT J */
         DD    =ABS(XVAL-X(I)),.
         IF DD LE D
         THEN DO,.
               D     =DD,.
               J     =I,.
               END,.
         END,.
   A,ARG(1)=X(J),.
   GO TO COM,.
ALIE..
/***********************************************************************/
/*                                                                     */
/*      AITKEN SCHEME FOR INTERPOLATION OF FUNCTION VALUE              */
/*      FROM GIVEN EQUIDISTANT TABLE                                   */
/*                                                                     */
/***********************************************************************/
   ENTRY (XST,DX,Y,DIM,ORDER,EPS,XVAL,YVAL),.
   SW    ='E',.
   Z1    =XST,.                        /*EQUIDISTANT ARGUMENTS        */
   Z2    =DX,.
   J     =1,.
   A,ARG(1)=Z1,.
   IF Z2= 0
   THEN GO TO COM,.
   J     =MAX(1,(XVAL-Z1)/Z2+1.5),.    /*COMPUTE STARTING SUBSCRIPT J */
   J     =MIN(DIM,J),.
   A,ARG(1)=Z1+FLOAT(J-1)*Z2,.
COM..
   ERROR='2',.
   XS    =XVAL,.
   DIMS =DIM,.
   N     =MIN(DIMS,ORDER),.
   DELT1,JL,JR=0,.
   VALI1,VAL(1)=Y(J),.
   FACT =XS-A,.
   DIST1=ABS(FACT),.
   N     =MAX(N,1),.
         DO I =2 TO N,.                /*TABLE SELECTION              */
         JJR   =J+JR,.                 /*TEST IF SUBSCRIPT IS GREATER */
         IF JJR GE DIMS                /*THAN DIM OR LESS THAN ONE    */
         THEN GO TO LAB2,.
         JJL   =J-JL,.
         IF JJL LE 1
         THEN GO TO LAB3,.
         IF SW= 'E'
         THEN A      =-FACT*Z2,.       /*A=(ARG(I-1)-XVAL)*DX         */
         ELSE A      =ABS(X(JJR+1)-XS)
              -ABS(X(JJL-1)-XS),.
         IF A LE 0                     /*TEST IF THE NEXT STEP IS TO  */
         THEN GO TO LAB3,.             /*THE RIGHT OR TO THE LEFT     */
LAB2..                                 /*STEP TO THE LEFT             */
         JL    =JL+1,.
         K     =J-JL,.
         GO TO CONT,.
LAB3..                                 /*STEP TO THE RIGHT            */
         JR    =JR+1,.
         K     =J+JR,.
CONT..
         IF SW= 'E'
         THEN A      =Z1+FLOAT(K-1)*Z2,.
         ELSE A      =X(K),.
         FACT =XS-A,.
         IF SW='M'
         THEN DO,.
               DIST =ABS(FACT),.
               IF DIST LT DIST1
               THEN GO TO IDENT,.      /*ARGUMENTS NOT MONOTONIC      */
               DIST1=DIST,.
               END,.
         ARG(I)=A,.
         VALI,VAL(I)=Y(K),.
               DO II      =1 TO I-1,.  /*COMPUTE VAL(I)               */
               ARGI =ARG(II),.
               H     =ARGI-A,.
               IF H =0
               THEN GO TO IDENT,.
               VALI =(VAL(II)*FACT-VALI
                     *(XS-ARGI))/H,.
               END,.
         DELT2=ABS(VALI-VALI1),.
         VALI1,VAL(I)=VALI,.
         IF I GT 2
         THEN DO,.
               IF DELT2 LE EPS         /*TEST ON ACCURACY             */
               THEN GO TO STOP,.
               IF I GE 5               /*SINGLE PRECISION VERSION  /*S*/
 /*            IF I GE 8               /*DOUBLE PRECISION VERSION  /*D*/
               THEN IF DELT2 GE DELT1  /*TEST ON OSCILLATION          */
               THEN GO TO OSCIL,.
               END,.
         DELT1=DELT2,.
         END,.                         /*END OF AITKEN-LOOP           */
   I     =N,.
   GO TO RETURN,.
```

```
OSCIL..
   ERROR='1',.
   GO TO IDENT1,.
IDENT..
   ERROR='3',.
IDENT1..
   I     =I-1,.
   GO TO RETURN,.
STOP..
   ERROR='0',.
RETURN..
   YVAL =VAL(I),.
   END,.                               /*END OF PROCEDURE ALI         */
```

Purpose:

ALIM interpolates the function value YVAL for a given argument value XVAL using a given table (X, Y) of argument and function values.

Usage:

CALL ALIM (X, Y, DIM, ORDER, EPS, XVAL, YVAL);

X - BINARY FLOAT [(53)]
Given vector of monotonic argument values.

Y - BINARY FLOAT [(53)]
Given vector of table-function values.

DIM - BINARY FIXED
Given dimension of vector X and Y.

ORDER - BINARY FIXED
Given number of points to be selected out of the given table (X, Y)

EPS - BINARY FLOAT [(53)]
Given constant used as upper bound for the absolute error.

XVAL - BINARY FLOAT [(53)]
Given argument to be interpolated.

YVAL - BINARY FLOAT [(53)]
Resultant interpolated function value.

Purpose:

ALIE interpolates the function value YVAL for a given argument value XVAL using XST, the starting value of the arguments, DX, the increment of the argument values, and the vector Y of function values.

Usage:

CALL ALIE (XST, DX, Y, DIM, ORDER, EPS, XVAL, YVAL);

XST - BINARY FLOAT [(53)]
Given starting value of arguments.

DX - BINARY FLOAT [(53)]
Given increment of argument values.

Y - BINARY FLOAT [(53)]
Given vector of table-function values.

DIM - BINARY FIXED
Given dimension of vector X and Y.

ORDER - BINARY FIXED
Given number of points to be selected out of the given table (X, Y).

EPS - BINARY FLOAT [(53)]
Given constant used as upper bound for the absolute error.

XVAL - BINARY FLOAT [(53)]
Given argument to be interpolated.

YVAL - BINARY FLOAT [(53)]
Resultant interpolated function value.

Remarks:

ERROR='0' - means required accuracy could be reached.

ERROR='1' - means required accuracy could not be reached because of rounding errors.

ERROR='2' - means accuracy could not be checked because MIN (DIM; ORDER) is less than 2, or the required accuracy could not be reached by means of the given table (X, Y). ORDER should be increased.

ERROR='3' - means two arguments in the argument vector X are identical, or the arguments are not monotonic.

In case ERROR='0' and ERROR='2' the last interpolated value for YVAL is returned. In case ERROR='1' and ERROR='3' the value prior to the last interpolated value for YVAL is returned. If, by a user error, ORDER is greater than DIM, the procedure selects only a maximum table of DIM points. In order to avoid errors, the user should check the correspondence between the selected table and its dimension by comparison of DIM and ORDER.

Method:

Interpolation is done by means of Aitken's scheme of Lagrange interpolation.

For reference see:

F.B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 49-50.

Mathematical Background:

Before starting Lagrange interpolation, a table (ARG, VAL) must be selected out of the given monotonic or equidistant table. This selection is done in two parts. In the first part, the subscript J of the argument next to the search argument XVAL is computed, using the following formulas:

In case of equidistant table -

$$\text{Subscript J = integer part of } \left(\frac{\text{XVAL-XST}}{\text{DX}} + 1.5\right)$$

In case of monotonic table -
Subscript J is searched for such that

$$\left|\text{XVAL - X(J)}\right| \le \left|\text{XVAL - X(I)}\right|, \; 1 \le \text{I} \le \text{DIM}$$

At each of the N = MIN (DIM, ORDER) interpolation steps, the procedure decides by comparison of distances whether the next step has to go to the right or to the left within the dimension of the given table.

It is assumed that | X(I) - XVAL|>|X(J) - XVAL| for all I >J. Otherwise, ERROR='3' is returned.

y_i means VAL(i); x_i means ARG(i).

Using the formulas

$$y_{i,n} = \frac{y_i\,(x_n - \text{XVAL}) - y_n\,(x_1 - \text{XVAL})}{x_n - x_1}$$

and

$$\begin{aligned} y_{1,2,\ldots,m,n} = {} & y_{1,2,\ldots,m}\,(x_n - \text{XVAL}) \\ & - y_{1,2,\ldots,m-1,n}\,(x_m - \text{XVAL}) \\ & / \,(x_n - x_m) \end{aligned}$$

it is possible to generate, by row, the following triangular Aitken scheme:

$$\begin{array}{llllll} x_1 & y_1 & & & & \\ x_2 & y_2 & y_{1,2} & & & \\ x_3 & y_3 & y_{1,3} & y_{1,2,3} & & \\ x_4 & y_4 & y_{1,4} & y_{1,2,4} & y_{1,2,3,4} & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_n & y_n & y_{1,n} & y_{1,2,n} & y_{1,2,3,n} \cdots & y_{1,2,3,\ldots,n} \end{array}$$

All resultant values of row I are stored in VAL(i):

$$\begin{aligned} \text{VAL(i)} = {} & \text{VAL(ii)} \cdot (\text{XVAL} - \text{ARG(i)}) \\ & - \text{VAL(i)}\ (\text{XVAL} - \text{ARG(ii)}) / (\text{ARG(ii)} - \text{ARG(i)}) \end{aligned}$$

(ii = 1,2,..., i - 1) for i = 2, 3, ..., MIN(DIM, ORDER).

Programming Considerations:

The procedure stops under the following conditions:

1. If the difference $|(VAL(i-1) - VAL(i))|$, with $i \geq 3$, of two successive values is less then a given tolerance EPS, ERROR='0' is returned.

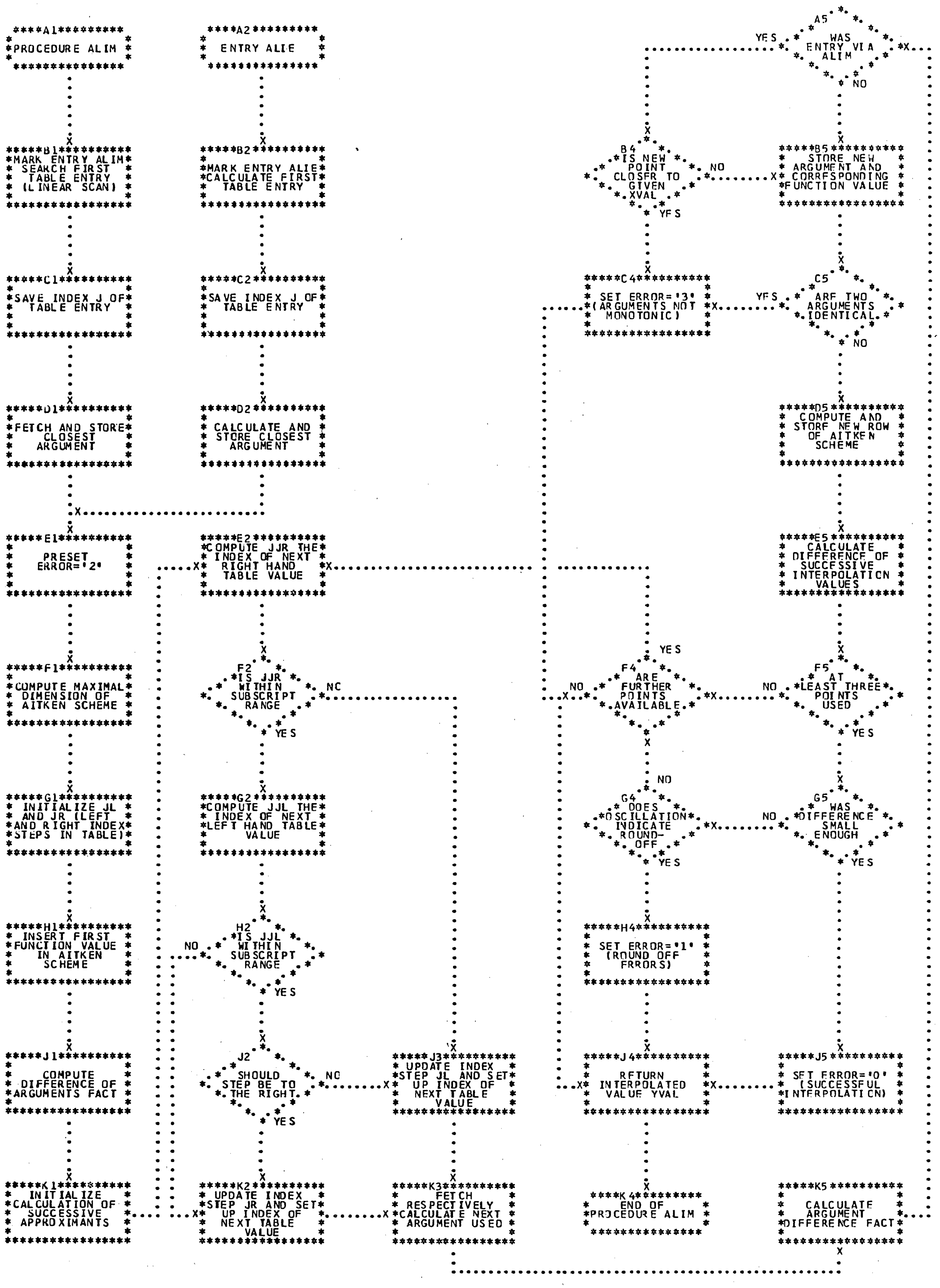
2. If the absolute value of this difference stops diminishing, thus showing the influence of rounding errors, ERROR='1' is returned. (Test starts at step i = 5 for single precision, step i = 8 for double precision.)

3. If the procedure has worked through the whole triangular Aitken scheme, ERROR='2' is returned.

4. If the procedure discovers that the arguments are not monotonic or that two arguments are identical, ERROR='3' is returned.

PROCEDURE ALIM USES AITKEN'S SCHEME FOR INTERPOLATION IN GIVEN MONOTONIC TABLE

ENTRY ALIE INTERPOLATES IN EQUIDISTANT TABLE



A1 PROCEDURE ALIM
A2 ENTRY ALIE
A5 WAS ENTRY VIA ALIM
YES
NO
B1 MARK ENTRY ALIM SEARCH FIRST TABLE ENTRY (LINEAR SCAN)
B2 MARK ENTRY ALIE CALCULATE FIRST TABLE ENTRY
B4 IS NEW POINT CLOSER TO GIVEN XVAL
NO
YES
B5 STORE NEW ARGUMENT AND CORRESPONDING FUNCTION VALUE
C1 SAVE INDEX J OF TABLE ENTRY
C2 SAVE INDEX J OF TABLE ENTRY
C4 SET ERROR='3' (ARGUMENTS NOT MONOTONIC)
C5 ARE TWO ARGUMENTS IDENTICAL
YES
NO
D1 FETCH AND STORE CLOSEST ARGUMENT
D2 CALCULATE AND STORE CLOSEST ARGUMENT
D5 COMPUTE AND STORE NEW ROW OF AITKEN SCHEME
E1 PRESET ERROR='2'
E2 COMPUTE JJR THE INDEX OF NEXT RIGHT HAND TABLE VALUE
E5 CALCULATE DIFFERENCE OF SUCCESSIVE INTERPOLATION VALUES
F1 COMPUTE MAXIMAL DIMENSION OF AITKEN SCHEME
F2 IS JJR WITHIN SUBSCRIPT RANGE
NO
YES
F4 ARE FURTHER POINTS AVAILABLE
YES
NO
F5 AT LEAST THREE POINTS USED
NO
YES
G1 INITIALIZE JL AND JR (LEFT AND RIGHT INDEX STEPS IN TABLE)
G2 COMPUTE JJL THE INDEX OF NEXT LEFT HAND TABLE VALUE
G4 DOES OSCILLATION INDICATE ROUND-OFF
NO
YES
G5 WAS DIFFERENCE SMALL ENOUGH
NO
YES
H1 INSERT FIRST FUNCTION VALUE IN AITKEN SCHEME
H2 IS JJL WITHIN SUBSCRIPT RANGE
NO
YES
H4 SET ERROR='1' (ROUND OFF ERRORS)
J1 COMPUTE DIFFERENCE OF ARGUMENTS FACT
J2 SHOULD STEP BE TO THE RIGHT
NO
YES
J3 UPDATE INDEX STEP JL AND SET UP INDEX OF NEXT TABLE VALUE
J4 RETURN INTERPOLATED VALUE YVAL
J5 SET ERROR='0' (SUCCESSFUL INTERPOLATION)
K1 INITIALIZE CALCULATION OF SUCCESSIVE APPROXIMANTS
K2 UPDATE INDEX STEP JR AND SET UP INDEX OF NEXT TABLE VALUE
K3 FETCH RESPECTIVELY CALCULATE NEXT ARGUMENT USED
K4 END OF PROCEDURE ALIM
K5 CALCULATE ARGUMENT DIFFERENCE FACT

Subroutine AHIM/AHIE

```
AHIM..                                                                  AHI   10
/**********************************************************************/AHI   20
/*                                                                    */AHI   30
/*      AITKEN HERMITE SCHEME FOR INTERPOLATION OF FUNCTION VALUE    */AHI   40
/*      FROM GIVEN MONOTONIC TABLE                                    */AHI   50
/*                                                                    */AHI   60
/**********************************************************************/AHI   70
   PROCEDURE (X,Y,DY,DIM,ORDER,EPS,XVAL,YVAL),.                         AHI   80
   DECLARE                                                              AHI   90
        (DIM,DIMS,I,II,J,JJL,JJR,JL,JR,K,N,ORDER)                       AHI  100
        BINARY FIXED,                                                   AHI  110
        (X(*),Y(*),DY(*),ARG(MIN(DIM,ORDER)),VAL(2*MIN(DIM,ORDER)),     AHI  120
        EPS,XVAL,YVAL,XST,DX,A,D,DD,DELT1,DELT2,DIST,DIST1,H,           AHI  130
        H1,H2,VALI,VALI1,VALJ,VALJ1,XS,Y1,YS,Z1,Z2)                     AHI  140
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION /*S*/AHI  150
 /*     BINARY FLOAT (53),            /*DOUBLE PRECISION VERSION /*D*/AHI  160
        (ERROR EXTERNAL,SW)                                             AHI  170
        CHARACTER(1),.                                                  AHI  180
   SW   ='M',.                        /*MONOTONIC ARGUMENTS           */AHI  190
   J    =1,.                                                            AHI  200
   D    =1E75,.                                                         AHI  210
        DO I = 1 TO DIM,.             /*COMPUTE STARTING SUBSCRIPT J */AHI  220
        DD   =ABS(XVAL-X(I)),.                                          AHI  230
        IF DD LE D                                                      AHI  240
        THEN DO,.                                                       AHI  250
             D    =DD,.                                                 AHI  260
             J    =I,.                                                  AHI  270
             END,.                                                      AHI  280
        END,.                                                           AHI  290
   ARG(1)=X(J),.                                                        AHI  300
   GO TO COM,.                                                          AHI  310
AHIE..                                                                  AHI  320
/**********************************************************************/AHI  330
/*                                                                    */AHI  340
/*      AITKEN HERMITE SCHEME FOR INTERPOLATION OF FUNCTION VALUE    */AHI  350
/*      FROM GIVEN EQUIDISTANT TABLE                                  */AHI  360
/*                                                                    */AHI  370
/**********************************************************************/AHI  380
   ENTRY (XST,DX,Y,DY,DIM,ORDER,EPS,XVAL,YVAL),.                        AHI  390
   SW   ='E',.                                                          AHI  400
   Z1   =XST,.                        /*EQUIDISTANT ARGUMENTS         */AHI  410
   Z2   =DX,.                                                           AHI  420
   J    =1,.                                                            AHI  430
   ARG(1)=Z1,.                                                          AHI  440
   IF Z2= 0                                                             AHI  450
   THEN GO TO COM,.                                                     AHI  460
   J    =MAX(1,(XVAL-Z1)/Z2+1.5),.    /*COMPUTE STARTING SUBSCRIPT J */AHI  470
   J    =MIN(DIM,J),.                                                   AHI  480
   ARG(1)=Z1+FLOAT(J-1)*Z2,.                                            AHI  490
COM..                                                                   AHI  500
   ERROR='2',.                                                          AHI  510
   XS   =XVAL,.                                                         AHI  520
   YS   =YVAL,.                                                         AHI  530
   DIMS =DIM,.                                                          AHI  540
   N    =MIN(DIMS,ORDER),.                                              AHI  550
   JL,JR=0,.                                                            AHI  560
   VALI,VAL(1)=Y(J),.                                                   AHI  570
   VALJ,VAL(2)=DY(J),.                                                  AHI  580
   H2   =XS-ARG(1),.                                                    AHI  590
   DIST1=ABS(H2),.                                                      AHI  600
   IF N LE 1                                                            AHI  610
   THEN DO,.                                                            AHI  620
        IF N = 1                                                        AHI  630
        THEN VAL(1)=VAL(I)+VAL(J)*H2,.                                  AHI  640
        ELSE VAL(1)=YS,.                                                AHI  650
        GO TO RETURN,.                                                  AHI  660
        END,.                                                           AHI  670
        DO I =2 TO N,.                /*TABLE SELECTION               */AHI  680
        JJR  =J+JR,.                                                    AHI  690
        IF JJR GE DIMS                                                  AHI  700
        THEN GO TO LAB2,.                                               AHI  710
        JJL  =J-JL,.                                                    AHI  720
        IF JJL LE 1                                                     AHI  730
        THEN GO TO LAB3,.                                               AHI  740
        IF SW= 'E'                                                      AHI  750
        THEN A    =(ARG(I-1)-XS)*Z2,. /*A=(ARG(I-1)-XVAL)*DX          */AHI  760
        ELSE A    =ABS(X(JJR+1)-XS)                                     AHI  770
             -ABS(X(JJL-1)-XS),.                                        AHI  780
        IF A LE 0                                                       AHI  790
        THEN GO TO LAB3,.                                               AHI  800
LAB2..                                                                  AHI  810
        JL   =JL+1,.                  /*STEP TO THE LEFT              */AHI  820
        K    =J-JL,.                                                    AHI  830
        GO TO CONT,.                                                    AHI  840
LAB3..                                                                  AHI  850
        JR   =JR+1,.                  /*STEP TO THE RIGHT             */AHI  860
        K    =J+JR,.                                                    AHI  870
CONT..                                                                  AHI  880
        IF SW= 'E'                                                      AHI  890
        THEN A    =Z1+FLOAT(K-1)*Z2,.                                   AHI  900
        ELSE DO,.                                                       AHI  910
             A    =X(K),.                                               AHI  920
             DIST =ABS(XS-A),.                                          AHI  930
             IF DIST LT DIST1                                           AHI  940
             THEN GO TO IDENT,.       /*ARGUMENTS NOT MONOTONIC       */AHI  950
             DIST1=DIST,.                                               AHI  960
             END,.                                                      AHI  970
        II   =I+I,.                                                     AHI  980
        VALJ1=DY(K),.                 /*VAL(2*I)=DY(K)                */AHI  990
        VALI1=Y(K),.                  /*VAL(2*I-1)=Y(K)               */AHI 1000
        ARG(I)=A,.                                                      AHI 1010
        VAL(II-3)=VALI+VALJ*H2,.                                        AHI 1020
        H1   =H2,.                                                      AHI 1030
        H2   =XS-A,.                                                    AHI 1040
        H    =H1-H2,.                                                   AHI 1050
        IF H = 0                                                        AHI 1060
        THEN GO TO IDENT,.            /*TWO IDENTICAL ARGUMENTS       */AHI 1070
        VAL(II-2)=VALI+(VALI1                                           AHI 1080
             -VALI)*H1/H,.                                              AHI 1090
        VALI =VALI1,.                                                   AHI 1100
        VALJ =VALJ1,.                                                   AHI 1110
        END,.                         /*END OF TABLE SELECTION        */AHI 1120
   VAL(II-1)=VALI+VALJ*H2,.                                             AHI 1130
   DELT2=0,.                          /*PREPARE AITKEN-SCHEME         */AHI 1140
   Y1   =VAL(1),.                                                       AHI 1150
        DO I = 1 TO N+N-2,.           /*START AITKEN-LOOP             */AHI 1160
        YS   =Y1,.                                                      AHI 1170
        DELT1=DELT2,.                                                   AHI 1180
        H1   =ARG((I+3)/2),.                                            AHI 1190
```

```
        Y1   =VAL(I+1),.                                                AHI 1200
             DO K = I TO 1 BY -1,.                                      AHI 1210
             H2   =ARG((K+1)/2),.                                       AHI 1220
             H    =H2-H1,.                                              AHI 1230
             IF H = 0                 /*COMPUTE DIAGONALS OF AITKEN- */AHI 1240
             THEN GO TO IDENT,.       /*SCHEME                        */AHI 1250
             Y1,VAL(K)=(VAL(K)*(XS-H1)                                  AHI 1260
                  -Y1*(XS-H2))/H,.                                      AHI 1270
             END,.                                                      AHI 1280
        DELT2=ABS(YS-Y1),.            /*TEST ON ACCURACY              */AHI 1290
        IF DELT2 LE EPS                                                 AHI 1300
        THEN GO TO STOP,.                                               AHI 1310
        IF I GE 5                     /*SINGLE PRECISION VERSION /*S*/AHI 1320
 /*     IF I GE 8                     /*DOUBLE PRECISION VERSION /*D*/AHI 1330
        THEN IF DELT2 GE DELT1                                          AHI 1340
        THEN GO TO OSCIL,.                                              AHI 1350
        END,.                         /*END OF AITKEN-LOOP            */AHI 1360
   GO TO RETURN,.                                                       AHI 1370
OSCIL..                               /*DELT2 STARTS OSCILLATING      */AHI 1380
   ERROR='1',.                                                          AHI 1390
   VAL(1)=YS,.                                                          AHI 1400
   GO TO RETURN,.                                                       AHI 1410
IDENT..                                                                 AHI 1420
   VAL(1)=YS,.                                                          AHI 1430
   ERROR='3',.                                                          AHI 1440
   GO TO RETURN,.                                                       AHI 1450
STOP..                                                                  AHI 1460
   ERROR='0',.                                                          AHI 1470
RETURN..                                                                AHI 1480
   YVAL =VAL(1),.                                                       AHI 1490
   END,.                              /*END OF PROCEDURE AHI          */AHI 1500
```

Purpose:

AHIM interpolates the function value YVAL for a given argument value XVAL using a given table (X, Y, DY) of argument values, function values, and their derivatives.

Usage:

CALL AHIM(X, Y, DY, DIM, ORDER, EPS, XVAL, YVAL);

X - BINARY FLOAT[(53)]
Given vector of monotonic arguments.

Y - BINARY FLOAT[(53)]
Given vector of table-function values.

DY - BINARY FLOAT [(53)]
Given vector of derivative values.

DIM - BINARY FIXED
Given dimension of vector X, Y, DY.

ORDER - BINARY FIXED
Given number of points to be selected out of the given table (X, Y, DY).

EPS - BINARY FLOAT [(53)]
Given constant used as upper bound for the absolute error.

XVAL - BINARY FLOAT [(53)]
Given argument to be interpolated.

YVAL - BINARY FLOAT [(53)]
Resultant interpolated function value.

Purpose:

AHIE interpolates the function value YVAL for a given argument value XVAL using XST, the starting value of the argument, DX, the increment of the argument values, vector Y of the function values, and vector DY of the function derivative values.

Usage:

CALL AHIE (XST, DX, Y, DY, DIM, ORDER, EPS, XVAL, YVAL);

XST - BINARY FLOAT [(53)]
Given starting value of the arguments.

DX - BINARY FLOAT [(53)]
Given increment of the argument values.

Y - BINARY FLOAT [(53)]
Given vector of table-function values.

DY - BINARY FLOAT [(53)]
Given vector of function derivative values.

DIM - BINARY FIXED
Given dimension of the vector X, Y, DY.

ORDER - BINARY FIXED
Given number of points to be selected out of the given table (X, Y, DY).

EPS - BINARY FLOAT [(53)]
Given constant used as the upper bound for the absolute error.

XVAL - BINARY FLOAT [(53)]
Given argument to be interpolated.

YVAL - BINARY FLOAT [(53)]
Resultant interpolated function value.

Remarks:

ERROR='0' means required accuracy could be reached.

ERROR='1' means required accuracy could not be reached because of rounding errors.

ERROR='2' means accuracy could not be checked because MIN(DIM, ORDER) is less than 2, or the required accuracy could not be reached by means of the given table (X, Y, DY). ORDER should be increased.

ERROR='3' means two arguments in argument vector X are identical or the arguments are not monotonic.

In the case ERROR='0' and ERROR='2' the last interpolated value of YVAL is returned. The value prior to the last interpolated value for YVAL is returned.

If, by a user error, ORDER is greater than DIM, the procedure selects only a maximum table of DIM points. In order to avoid errors, the user should check the correspondence between the selected table and its discussion by comparison of DIM and ORDER.

Method:

Interpolation is done by means of Aitken's scheme of Hermite interpolation.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, 11. 314-317.

Gershinsky and Levine, "Aitken-Hermite Interpolation" JACM, vol. 11, issue 3 (1964), pp. 352-356.

Mathematical Background:

Before starting Hermite interpolation, a table (ARG, VAL) must be selected out of the given monotonic or equidistant table. This selection is done in two parts. In the first part, the subscript J of the argument next to the search argument XVAL is computed, using the following formulas:

In case of the equidistant table -

Subscript J = the integer part of

$$\left(\frac{XVAL-XST}{DX} + 1.5 \right)$$

In case of the monotonic table -

Subscript J is searched for such that

$$\left| XVAL - X(J) \right| \leq \left| XVAL - X(I) \right| , \; 1 \leq I \leq DIM$$

At each of the N = MIN(DIM, ORDER) selection steps, the procedure decides, by comparison of distances, whether the next step in vector X has to go to the right or to the left within the dimension of the given table, and replaces the components of vector VAL (that is, function and derivative values) by interpolation values Z_i of the first order (see Figure 8, third column). This is done by the following formulas:

$$VAL(i) = y_i + VAL\ (i + 1) \cdot H1 \quad (i=1, 3, \ldots, 2n-1)$$

$$VAL(i+1) = y_i + (VAL(i+2) - y) \cdot \frac{H1}{H1-H2} \ (i=1, 3, \ldots,$$

2n-3)

with

n = MIN(DIM, ORDER), y_i = VAL(i)

$H1 = XVAL - ARG(j-1)$, $H2 = XVAL - ARG(j)$

and

$$j = \frac{i+1}{2} + 1$$

Now it is possible to generate successively the upward diagonals of the triangular Aitken scheme, using the following formulas:

$$z_{1,2} = \frac{1}{x_2 - x_1} \cdot \begin{vmatrix} z_1 & x_1 - XVAL \\ z_2 & x_2 - XVAL \end{vmatrix}$$

$$z_{2,3} = \frac{1}{x_2 - x_1} \cdot \begin{vmatrix} z_2 & x_1 - XVAL \\ z_3 & x_2 - XVAL \end{vmatrix}$$

$$z_{1,2,3} = \frac{1}{x_2 - x_1} \cdot \begin{vmatrix} z_{1,2} & x_1 - XVAL \\ z_{2,3} & x_2 - XVAL \end{vmatrix}$$

$$z_{3,4} = \frac{1}{x_3 - x_2} \cdot \begin{vmatrix} z_3 & x_2 - XVAL \\ z_4 & x_3 - XVAL \end{vmatrix}$$

with

$$x_i = ARG\ (i).$$

All resultant values of an upward diagonal can be stored in positions of vector VAL with decreasing subscripts: $VAL(k) =$

$$\frac{VAL(k) \cdot (XVAL - H1) - VAL(k+1) \cdot (XVAL - ARG(l))}{ARG(l) - H1}$$

for

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

where

$$k = i-j+1, \quad m = \left[\frac{i+3}{2}\right],$$

$$l = \left[\frac{k+1}{2}\right]$$

and $H1 = ARG\ (m)$

for $i = 1, 2, \ldots, 2n-2$.

$ARG(1) = x_1$	$VAL(1) = y_1$	$VAL(1) = z_1$	$z_{1,2}$	$z_{1,2,3}$	$z_{1,2,3,4}$..
	$VAL(2) = y'_1$	$VAL(2) = z_2$	$z_{2,3}$	$z_{2,3,4}$	.
					.
$ARG(2) = x_2$	$VAL(3) = y_2$	$VAL(3) = z_3$	$z_{3,4}$	$z_{3,4,5}$	.
.	$VAL(4) = y'_2$	$VAL(4) = z_4$	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
$ARG(n) = x_n$	$VAL(2n-1) = y_n$	$VAL(2n-1)$			
	$VAL(2n) = y'_n$	$= \quad z_{2n-1}$			

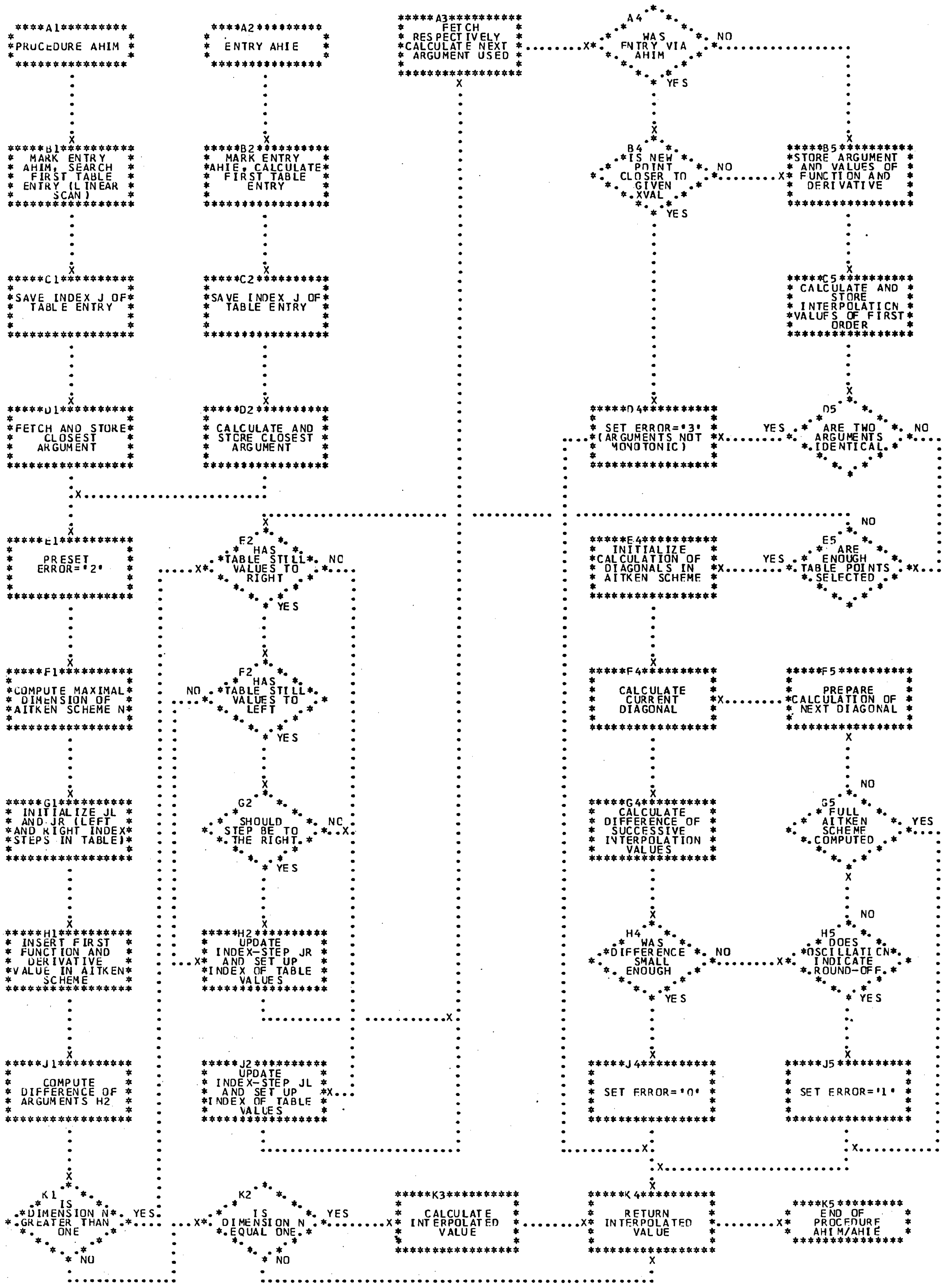
Figure 8. Triangular scheme for Aitken-Hermite interpolation

Programming Considerations

The procedure stops under the following conditions:

1. If the absolute value of the difference between two successive interpolated values VAL(1) is less than a given tolerance EPS, ERROR='0' is returned.
2. If the absolute value of this difference stops diminishing (thus showing the influence of rounding errors), ERROR='1' is returned. (Test starts at step i = 5 for single precision, i = 8 for double precision.)
3. If the procedure has worked through the whole triangular scheme, ERROR='2' is returned (see "Remarks", above).
4. If the procedure discovers two table points with identical arguments or the arguments are not monotonic, ERROR='3' is returned.

PROCEDURE AHIM USES AITKEN-HERMITE SCHEME FOR INTERPOLATION IN GIVEN MONOTONIC TABLE
ENTRY AHIE INTERPOLATES IN EQUIDISTANT TABLE


A1 PROCEDURE AHIM
A2 ENTRY AHIE
A3 FETCH RESPECTIVELY CALCULATE NEXT ARGUMENT USED
A4 WAS ENTRY VIA AHIM
NO
YES
B1 MARK ENTRY AHIM, SEARCH FIRST TABLE ENTRY (LINEAR SCAN)
B2 MARK ENTRY AHIE, CALCULATE FIRST TABLE ENTRY
B4 IS NEW POINT CLOSER TO GIVEN XVAL
NO
YES
B5 STORE ARGUMENT AND VALUES OF FUNCTION AND DERIVATIVE
C1 SAVE INDEX J OF TABLE ENTRY
C2 SAVE INDEX J OF TABLE ENTRY
C5 CALCULATE AND STORE INTERPOLATION VALUES OF FIRST ORDER
D1 FETCH AND STORE CLOSEST ARGUMENT
D2 CALCULATE AND STORE CLOSEST ARGUMENT
D4 SET ERROR='3' (ARGUMENTS NOT MONOTONIC)
D5 ARE TWO ARGUMENTS IDENTICAL
YES
NO
E1 PRESET ERROR='2'
E2 HAS TABLE STILL VALUES TO RIGHT
NO
YES
E4 INITIALIZE CALCULATION OF DIAGONALS IN AITKEN SCHEME
E5 ARE ENOUGH TABLE POINTS SELECTED
YES
NO
F1 COMPUTE MAXIMAL DIMENSION OF AITKEN SCHEME N
F2 HAS TABLE STILL VALUES TO LEFT
NO
YES
F4 CALCULATE CURRENT DIAGONAL
F5 PREPARE CALCULATION OF NEXT DIAGONAL
G1 INITIALIZE JL AND JR (LEFT AND RIGHT INDEX STEPS IN TABLE)
G2 SHOULD STEP BE TO THE RIGHT.
NO
YES
G4 CALCULATE DIFFERENCE OF SUCCESSIVE INTERPOLATION VALUES
G5 FULL AITKEN SCHEME COMPUTED
NO
YES
H1 INSERT FIRST FUNCTION AND DERIVATIVE VALUE IN AITKEN SCHEME
H2 UPDATE INDEX-STEP JR AND SET UP INDEX OF TABLE VALUES
H4 WAS DIFFERENCE SMALL ENOUGH
NO
YES
H5 DOES OSCILLATION INDICATE ROUND-OFF.
NO
YES
J1 COMPUTE DIFFERENCE OF ARGUMENTS H2
J2 UPDATE INDEX-STEP JL AND SET UP INDEX OF TABLE VALUES
J4 SET ERROR='0'
J5 SET ERROR='1'
K1 IS DIMENSION N GREATER THAN ONE
YES
NO
K2 IS DIMENSION N EQUAL ONE.
YES
NO
K3 CALCULATE INTERPOLATED VALUE
K4 RETURN INTERPOLATED VALUE
K5 END OF PROCEDURE AHIM/AHIE

Subroutine ACFM/ACFE

```
ACFM..
/**********************************************************************/
/*                                                                    */
/*      CONTINUED FRACTION SCHEME FOR INTERPOLATION OF FUNCTION VALUE*/
/*      FROM GIVEN MONOTONIC TABLE                                    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (X,Y,DIM,ORDER,EPS,XVAL,YVAL),.
   DECLARE
        (DIM,I,J,K,N,II,III,JL,JR,JJL,JJR,DIMS,ORDER)
        BINARY FIXED,
        (X(*),Y(*),ARG(MIN(DIM,ORDER)),VAL(MIN(DIM,ORDER)),XVAL,A1,
        YVAL,XST,DX,EPS,XS,Z1,Z2,D,DD,VALI,ARGI,A,DIST,DIST1,H,DELT1,
        DELT2,ARGJ,P1,P2,P3,Q1,Q2,Q3,ZS,YS,ARGI1,VALI1,EPS1)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY'FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        (ERROR EXTERNAL,SW)
        CHARACTER (1),.
   SW   ='M',.                         /*MONOTONIC ARGUMENTS           */
   J    =1,.
   D    =1E75,.
        DO I = 1 TO DIM,.
        DD   =ABS(XVAL-X(I)),.
        IF DD LE D
        THEN DO,.
             D    =DD,.
             J    =I,.
             END,.
        END,.
   ARGI,ARG(1)=X(J),.
   GO TO COM,.
ACFE..
/**********************************************************************/
/*                                                                    */
/*      CONTINUED FRACTION SCHEME FOR INTERPOLATION OF FUNCTION VALUE*/
/*      FROM GIVEN EQUIDISTANT TABLE                                  */
/*                                                                    */
/**********************************************************************/
   ENTRY (XST,DX,Y,DIM,ORDER,EPS,XVAL,YVAL),.
   SW   ='E',.
   Z1   =XST,.
   Z2   =DX,.
   J    =1,.
   ARGI,ARG(1)=Z1,.
   IF Z2= 0
   THEN GO TO COM,.
   J    =MAX(1,(XVAL-Z1)/Z2+1.5),.     /*COMPUTE STARTING SUBSCRIPT J */
   J    =MIN(DIM,J),.
   ARGI,ARG(1)=Z1+FLOAT(J-1)*Z2,.
COM..
   EPS1 =1E-6,.                        /*SINGLE PRECISION VERSION  /*S*/
 /*EPS1 =1E-13,.                       /*DOUBLE PRECISION VERSION  /*D*/
   ERROR='2',.
   XS   =XVAL,.
   DIMS =DIM,.
   N    =MIN(DIMS,ORDER),.
   Q2,DELT2,JL,JR=0,.
   P3,YS,VAL(1)=Y(J),.
   P2,Q3=1,.
   A1   =XS-ARGI,.
   DIST1=ABS(A1),.
        DO I = 2 TO N,.                /*START TABLE SELECTION         */
        JJR  =J+JR,.
        IF JJR GE DIMS                 /*TABLE SELECTION               */
        THEN GO TO LAB2,.
        JJL  =J-JL,.
        IF JJL LE 1
        THEN GO TO LAB3,.
        IF SW= 'E'
        THEN A    =-A1*Z2,.            /*A=(ARG(I-1)-XVAL)*DX          */
        ELSE A    =ABS(X(JJR+1)
             -XS)-ABS(X(JJL
             -1)-XS),.
        IF A LE 0
        THEN GO TO LAB3,.
LAB2..
        JL   =JL+1,.                   /*STEP TO THE LEFT              */
        K    =J-JL,.
        GO TO CONT,.
LAB3..
        JR   =JR+1,.                   /*STEP TO THE RIGHT             */
        K    =J+JR,.
CONT..
        IF SW= 'E'
        THEN A    =Z1+FLOAT(K-1)*Z2,.
        ELSE A    =X(K),.
        A1   =XS-A,.
        IF SW='M'
        THEN DO,.
             DIST =ABS(A1),.
             IF DIST LT DIST1
             THEN GO TO IDENT,.        /*ARGUMENTS NOT MONOTONIC       */
             DIST1=DIST,.
             END,.
        ARG(I)=A,.
        VAL(I)=Y(K),.
        END,.                          /*END OF TABLE SELECTION        */
   A1   =XS-ARG(1),.
        DO I = 2 TO N,.                /*START INTERPOLATION LOOP      */
        II   =0,.
        P1   =P2,.                     /*MOVE PARAMETERS P2,P3,Q2,Q3   */
        Q1   =Q2,.
        P2   =P3,.
        Q2   =Q3,.
        ZS   =YS,.
        DELT1=DELT2,.
        ARGI =ARG(I),.
        VALI =VAL(I),.
INVERT..                               /*COMPUTE INVERTED DIFFERENCES */
        ARGI1=ARGI,.
        VALI1=VALI,.
             DO J = 1 TO I-1,.
             ARGJ =ARG(J),.
             H    =VALI-VAL(J),.
             IF ABS(H) LE ABS(VALI)*EPS1
             THEN DO,.
                  IF ARGI= ARGJ        /*ERROR RETURNS,IF TWO          */
                  THEN GO TO IDENT,.   /*IDENTICAL ARGUMENTS EXIST     */
                  IF J GE I-1
                  THEN DO,.
                       II   =II+1,.    /*INTERCHANGE ROW I WITH        */
                       III  =I+II,.    /*ROW I+II                      */
                       IF III GT N
                       THEN GO TO RETURN,.
                       VALI =VAL(III),.
                       VAL(III)=VALI1,.
                       ARGI =ARG(III),.
                       ARG(III)=ARGI1,.
                       GO TO INVERT,.
                       END,.
                  VALI =1E75,.         /*VAL(I) = VAL(J), J LT I-1     */
                  END,.
             ELSE VALI =(ARGI          /*VAL(I) NE VAL(J)              */
                  -ARGJ)/H,.
             END,.
        P3   =VALI*P2+A1*P1,.          /*COMPUTE INVERTED DIFFERENCES */
        Q3   =VALI*Q2+A1*Q1,.          /*BY WALLIS-EULER SCHEME        */
        VAL(I)=VALI,.                  /*GENERATE NEW VAL(I),ARG(I)    */
        ARG(I)=ARGI,.
        A1   =XS-ARGI,.
        IF Q3= 0
        THEN YS   =1E75,.              /*Q3 = 0                        */
        ELSE YS   =P3/Q3,.             /*Q3 NE 0                       */
        DELT2=ABS(ZS-YS),.
        IF DELT2 LE EPS                /*TEST ON ACCURACY              */
        THEN GO TO STOP,.
        IF I GE 8                      /*SINGLE PRECISION VERSION  /*S*/
 /*     IF I GE 10                     /*DOUBLE PRECISION VERSION  /*D*/
        THEN IF DELT2 GE DELT1
        THEN GO TO OSCIL,.
        END,.                          /*END OF INTERPOLATION LOOP     */
   GO TO RETURN,.
IDENT..                                /*ARG(I) = ARG(J) FOR I NE J    */
   ERROR='3',.
   GO TO RETURN,.
OSCIL..                                /*DELT2 STARTS OSCILLATING      */
   YS   =ZS,.
   ERROR='1',.
   GO TO RETURN,.
STOP..
   ERROR='0',.
RETURN..
   YVAL =YS,.
   END,.                               /*END OF PROCEDURE ACFI         */
```

Purpose:

ACFM interpolates the function value YVAL for a given argument value XVAL using a given table (X, Y) of arguments and function values.

Usage:

CALL ACFM (X, Y, DIM, ORDER, EPS, XVAL, YVAL);

X - BINARY FLOAT [(53)]
Given vector of monotonic arguments.

Y - BINARY FLOAT [(53)]
Given vector table-function values.

DIM - BINARY FIXED
Given dimension of vector X and Y.

ORDER - BINARY FIXED
Given number of points to be selected out of the given table (X, Y).

EPS - BINARY FLOAT[(53)]
Given constant used as upper bound for the absolute error.

XVAL - BINARY FLOAT [(53)]
Given argument to be interpolated.

YVAL - BINARY FLOAT [(53)]
Resultant interpolated function value.

Purpose:

ACFE interpolates the function value YVAL for a given argument value XVAL using XST, the starting value of the arguments, DX, the increment of the argument values, and vector Y of function values.

Usage:

CALL ACFE (XST, DX, Y, DIM, ORDER, EPS, XVAL, YVAL);

XST - BINARY FLOAT [(53)]
Given the starting value of the arguments.

DX - BINARY FLOAT [(53)]
Given increment of the argument values.

Y - BINARY FLOAT [(53)]
Given vector of table-function values.

DIM - BINARY FIXED
Given dimension of vector X and Y.

ORDER - BINARY FIXED
Given number of points to be selected out of the given table (X, Y).

EPS - BINARY FLOAT [(53)]
Given constant used as upper bound for the absolute error.

XVAL - BINARY FLOAT [(53)]
Given argument to be interpolated.

YVAL - BINARY FLOAT [(53)]
Resultant interpolated function value.

Remarks:

See AHIM/AHIE, ALIM, ALIE

Method:

Interpolation is done by a continued fraction and inverted differences scheme.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 395-406.

Mathematical Background:

Before starting continued fraction interpolation, a table (ARG, VAL) must be selected out of the given monotonic or equidistant table. This selection is done before the continued fraction interpolation in the same way as in ALIM/ALIE.

It is assumed that $| x(i) - XVAL | > | x(j) - XVAL |$ for all $i > j$; otherwise, ERROR='3' is returned.

Using the following formulas:

$$y_{1,n} = \frac{x_n - x_1}{y_n - y_1}$$

$$y_{1,2,\dots,m,n} = \frac{x_n - x_m}{y_{1,2,\dots,m-1,n} - y_{1,2,\dots,m}}$$

with x_i = ARG(i), y_i = VAL(i)

the triangular scheme of inverted differences shown in Figure 9 can be generated by row for the table (ARG, VAL). All resultant values of row i can be stored in VAL(i). Thus, it is possible to generate the downward diagonal of the inverted differences scheme in vector VAL:

$$\text{VAL(i)} = \frac{\text{ARG(i)} - \text{ARG(j)}}{\text{VAL(i)} - \text{VAL(j)}} (j = 1, 2, \dots, i-1)$$

for i = 2, 3, . . . , MIN(DIM, ORDER).

If for j = i-1, VAL(i) is equal to the infinity element, table point ARG(i), VAL(i) is interchanged with a table point ahead.

Now, after computation of each new component VAL(i), continued fraction interpolation generates the following parameters using Wallis-Euler formula:

$$P3 = VAL(i) \cdot P2 + (XVAL-ARG(i-1)) \cdot P1$$

$$Q3 = VAL(i) \cdot Q2 + (XVAL-ARG(i-1)) \cdot Q1$$

and YVAL = P3/Q3,

starting with P1 = 1, P2 = VAL(1), Q1 = 0, Q2 = 1. After each step, P1 = P2, P2 = P3, Q1 = Q2, Q2 = Q3 are set.

ARG(1) = x_1	VAL(1) = y_1			
ARG(2) = x_2	VAL(2) = y_2	$\underline{y_{1,2}}$		
ARG(3) = x_3	VAL(3) = y_3	$y_{1,3}$	$\underline{y_{1,2,3}}$	
.	.	.	.	
.	.	.	.	
.	.	.	.	
.	.	.	.	
.	.	.	.	
ARG(n) = x_n	VAL(n) = y_n	$y_{1,n}$	$y_{1,2,n}$	$y_{1,2,3,\dots,n}$

Figure 9. Triangular scheme for fraction interpolation

Programming Considerations:

The procedure stops under the following conditions:

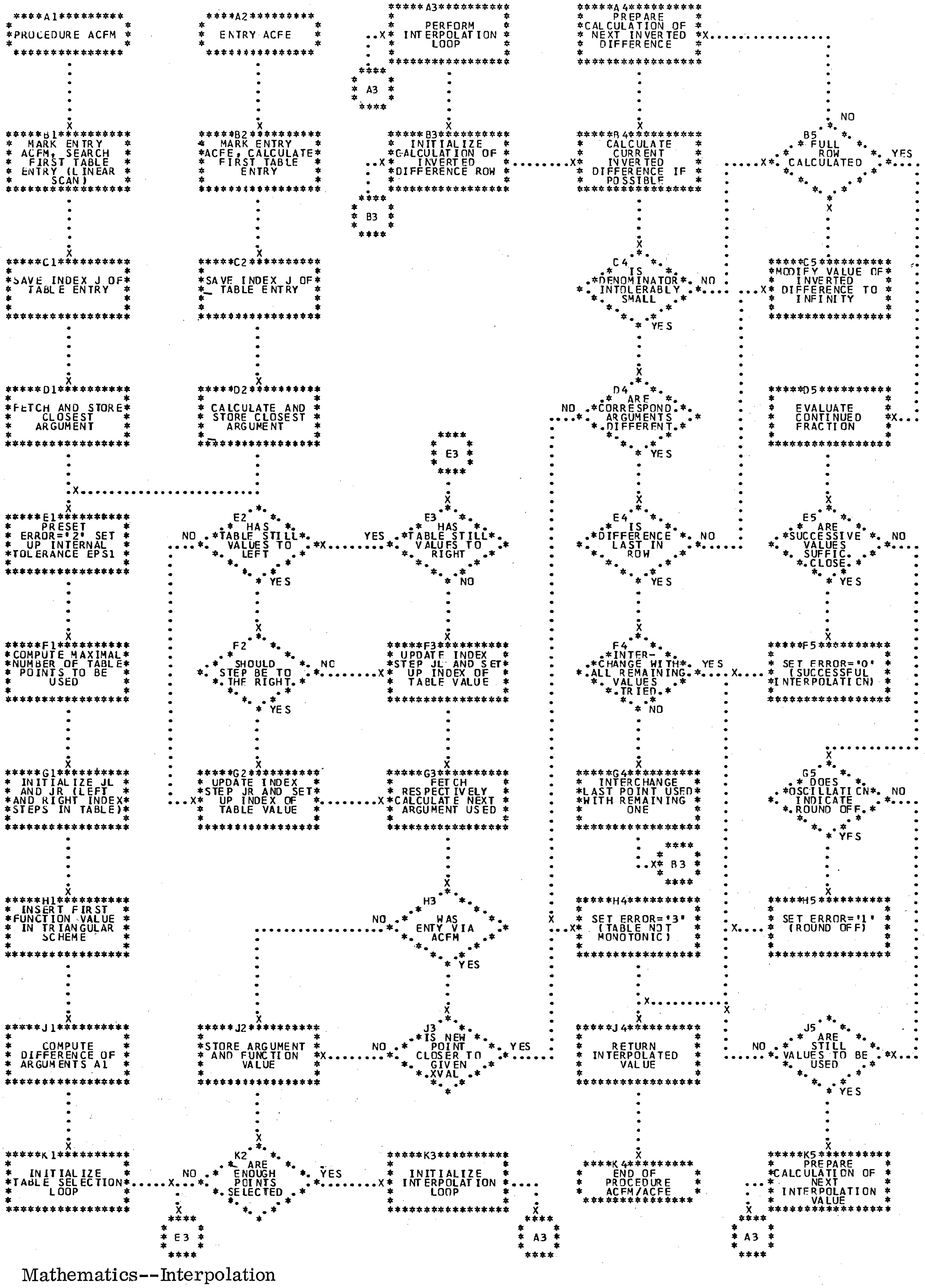
1. If the absolute value of the difference between two successive values of YVAL is less than a given tolerance EPS, ERROR='0' is returned.

2. If the absolute value of this difference starts oscillating, ERROR='1' is returned. (Test starts at step i = 8 for single precision, i = 10 for double precision.)

3. If the number of interpolation steps has become MIN(DIM, ORDER), ERROR='2' is returned.

4. If the procedure discovers that two table points have identical argument values or that the arguments are not monotonic, ERROR='3' is returned.

PROCEDURE ACFM PERFORMS CONTINUED FRACTICN INTERPCLATION IN A GIVEN MONOTONIC TABLE
ENTRY ACFE INTERPCLATES IN AN EQUIDISTANT TABLE


A1 PROCEDURE ACFM
A2 ENTRY ACFE
A3 PERFORM INTERPOLATION LOOP
A4 PREPARE CALCULATION OF NEXT INVERTED DIFFERENCE
A3
B1 MARK ENTRY ACFM, SEARCH FIRST TABLE ENTRY (LINEAR SCAN)
B2 MARK ENTRY ACFE, CALCULATE FIRST TABLE ENTRY
B3 INITIALIZE CALCULATION OF INVERTED DIFFERENCE ROW
B4 CALCULATE CURRENT INVERTED DIFFERENCE IF POSSIBLE
B5 FULL ROW CALCULATED
NO
YES
B3
C1 SAVE INDEX J OF TABLE ENTRY
C2 SAVE INDEX J OF TABLE ENTRY
C4 IS DENOMINATOR INTOLERABLY SMALL
NO
YES
C5 MODIFY VALUE OF INVERTED DIFFERENCE TO INFINITY
D1 FETCH AND STORE CLOSEST ARGUMENT
D2 CALCULATE AND STORE CLOSEST ARGUMENT
D4 ARE CORRESPONDING ARGUMENTS DIFFERENT
NO
YES
D5 EVALUATE CONTINUED FRACTION
E3
E1 PRESET ERROR='2' SET UP INTERNAL TOLERANCE EPS1
E2 HAS TABLE STILL VALUES TO LEFT
NO
YES
E3 HAS TABLE STILL VALUES TO RIGHT
YES
NO
E4 IS DIFFERENCE LAST IN ROW
NO
YES
E5 ARE SUCCESSIVE VALUES SUFFIC. CLOSE
NO
YES
F1 COMPUTE MAXIMAL NUMBER OF TABLE POINTS TO BE USED
F2 SHOULD STEP BE TO THE RIGHT
NO
YES
F3 UPDATE INDEX STEP JL AND SET UP INDEX OF TABLE VALUE
F4 INTER-CHANGE WITH ALL REMAINING VALUES TRIED
YES
NO
F5 SET ERROR='0' (SUCCESSFUL INTERPOLATION)
G1 INITIALIZE JL AND JR (LEFT AND RIGHT INDEX STEPS IN TABLE)
G2 UPDATE INDEX STEP JR AND SET UP INDEX OF TABLE VALUE
G3 FETCH RESPECTIVELY CALCULATE NEXT ARGUMENT USED
G4 INTERCHANGE LAST POINT USED WITH REMAINING ONE
G5 DOES OSCILLATION INDICATE ROUND OFF
NO
YES
B3
H1 INSERT FIRST FUNCTION VALUE IN TRIANGULAR SCHEME
H3 WAS ENTY VIA ACFM
NO
YES
H4 SET ERROR='3' (TABLE NOT MONOTONIC)
H5 SET ERROR='1' (ROUND OFF)
J1 COMPUTE DIFFERENCE OF ARGUMENTS A1
J2 STORE ARGUMENT AND FUNCTION VALUE
J3 IS NEW POINT CLOSER TO GIVEN XVAL
NO
YES
J4 RETURN INTERPOLATED VALUE
J5 ARE STILL VALUES TO BE USED
NO
YES
K1 INITIALIZE TABLE SELECTION LOOP
K2 ARE ENOUGH POINTS SELECTED
NO
YES
K3 INITIALIZE INTERPOLATION LOOP
K4 END OF PROCEDURE ACFM/ACFE
K5 PREPARE CALCULATION OF NEXT INTERPOLATION VALUE
E3
A3
A3

• Subroutine FFT

```
FFT..                                                                FFT   10
/*******************************************************************/FFT   20
/*                                                                 */FFT   30
/*      FAST  FOURIER  TRANSFORM  FOR ANY ONE-DIMENSIONAL ARRAY    */FFT   40
/*                                                                 */FFT   50
/*******************************************************************/FFT   60
   PROCEDURE(A,M,OPT),.                                              FFT   70
   DECLARE                                                           FFT   80
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR   */FFT   90
        (OPT,COPT) CHARACTER(1),                                     FFT  100
        (DA,DB,DC,DH,DS,RI)                                          FFT  110
        BINARY FLOAT(53),                                            FFT  120
        (A(*),S(2**(M-2)+1),AAR,                                     FFT  130
        AAI,ABR,ABI,AW,CO,SI)                                        FFT  140
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION /*S*/FFT  150
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION /*D*/FFT  160
        (I,ID,IND,IR,IST,                                            FFT  170
        J,K,L,M,N,NH,NQ)                                             FFT  180
        BINARY FIXED,.                                               FFT  190
   IF M LT 2                          /*TEST SPECIFIED DIMENSION M */FFT  200
   THEN DO ,.                                                        FFT  210
        ERROR='P',.                   /*P MEANS WRONG PARAMETEK    */FFT  220
        GO TO RETURN,.                                               FFT  230
        END,.                                                        FFT  240
   ERROR='C',.                        /*PRESET ERROR INDICATOR     */FFT  250
   COPT =OPT,.                                                       FFT  260
   N    =2**M,.                       /*INITIALIZE PARAMETERS      */FFT  270
   NH   =N/10B,.                                                     FFT  280
   NQ   =N/100B+2,.                                                  FFT  290
   L    =NQ+1,.                                                      FFT  300
   RI   =3.141592653589793E+00/NH,.   /*RI MEANS  2*PI/N           */FFT  310
   DA,S(1)=0,.                        /*SET SINE FOR 0 AND PI/2    */FFT  320
   DB,S(NQ-1)=1,.                                                    FFT  330
   DS,S(2)=SIN(RI),.                                                 FFT  340
   DC   =COS(RI),.                    /****************************/FFT  350
        DO I =3 TÓ N/1000B+1,.        /*CALCULATE SINE TERMS       */FFT  360
        RI   =DC*DB,.                 /*BETWEEN  0  AND  PI/2      */FFT  370
        S(L-I),DH=RI-DA,.             /****************************/FFT  380
        DA   =DB,.                                                   FFT  390
        DB   =RI+DH,.                 /*CALCULATION IS DONE USING  */FFT  400
        S(I) =DB*DS,.                 /*DOUBLE PRECISION ARITHMETIC */FFT  410
        END,.                                                        FFT  420
   IF COPT= '2'                       /*'2' MEANS CALCULATION OF   */FFT  430
   THEN GO TO REAL,.                  /*REAL FOURIER SERIES        */FFT  440
   IF COPT= '3'                       /*'3' MEANS CALCULATION OF   */FFT  450
   THEN GO TO INV,.                   /*COMPLEX FOURIER SERIES     */FFT  460
   AW   =1/NH,.                                                      FFT  470
        DO I =1 TO N,.                /*PREPARE VECTOR A FOR FINITE */FFT  480
        A(I) =A(I)*AW,.               /*FOURIER TRANSFORM          */FFT  490
        END,.                         /****************************/FFT  500
INV..                                 /*REORDER INITIAL TERMS A(I) */FFT  510
   J    =1,.                          /*BY BIT REVERSAL TECHNIQUE  */FFT  520
        DO I =1 TO N BY 2,.           /****************************/FFT  530
        IF J GT I                     /*IS BIT REVERSAL GREATER THAN */FFT  540
        THEN DO,.                     /*INIT. BINARY REPRESENTATION */FFT  550
             AAR  =A(J),.                                            FFT  560
             AAI  =A(J+1),.           /*INTERCHANGE A(I) WITH A(J) */FFT  570
             A(J) =A(I),.             /*AND  A(I+1)  WITH  A(J+1)  */FFT  580
             A(J+1)=A(I+1),.                                         FFT  590
             A(I) =AAR,.                                             FFT  600
             A(I+1)=AAI,.                                            FFT  610
             END,.                                                   FFT  620
        K    =NH,.                                                   FFT  630
             DO WHILE (J GT K),.      /*UPDATE  J  AND  K          */FFT  640
             J    =J-K,.                                             FFT  650
             K    =K/10B,.                                           FFT  660
             END,.                                                   FFT  670
        J    =J+K,.                   /*COMPUTE NEW BIT REVERSAL   */FFT  680
        END,.                                                        FFT  690
   IR,I =2,.                                                         FFT  700
   ID   =NH,.                         /****************************/FFT  710
CPLX..                                /*COMPLEX FOURIER TRANSFORM  */FFT  720
                                      /*WITH  N/2  ELEMENTS        */FFT  730
   IST  =I+I,.                        /****************************/FFT  740
   IND  =1,.                                                         FFT  750
        DO J =1 TO I BY 2,.                                          FFT  760
        SI   =-S(IND),.               /*STORE SINE VALUES IN SI    */FFT  770
        IF COPT= '3'                  /*CHANGE SIGN IN CASE OF     */FFT  780
        THEN SI   =-SI,.              /*FOURIER SERIES             */FFT  790
        CO   =S(NQ-IND),.             /*STORE COSINE VALUES IN CO  */FFT  800
        IF J GE IR                                                   FFT  810
        THEN DO,.                     /*MODIFY INDEX IND OF THE    */FFT  820
             IND  =IND-ID,.           /*SINE VECTOR S              */FFT  830
             CO   =-CO,.              /*COS(PI/2+B) = -SIN(B)      */FFT  840
             END,.                                                   FFT  850
        ELSE IND  =IND+ID,.                                          FFT  860
                                      /*EXECUTE TRANSFORMATION-LOOP */FFT  870
             DO K =J TO N BY IST,.                                   FFT  880
             L    =K+I,.                                             FFT  890
             AAR  =CO*A(L)-SI*A(L+1),.                               FFT  900
             AAI  =CO*A(L+1)+SI*A(L),.                               FFT  910
             A(L) =A(K)-AAR,.         /*MODIFY AND RESTORE ELEMENTS */FFT  920
             A(L+1)=A(K+1)-AAI,.                                     FFT  930
             A(K) =A(K)+AAR,.                                        FFT  940
             A(K+1)=A(K+1)+AAI,.                                     FFT  950
             END,.                                                   FFT  960
        END,.                                                        FFT  970
   IR   =I+1,.                        /*UPTATE PARAMETERS          */FFT  980
   I    =IST,.                                                       FFT  990
   ID   =ID/10B,.                                                    FFT 1000
   IF I LE NH                                                        FFT 1010
   THEN GO TB CPLX,.                  /*END OF OUTER LOOP          */FFT 1020
   IF COPT= '1'                       /*'1'  AND  '3'  MEAN COMPLEX */FFT 1030
   THEN GO TO RETURN,.                /*FOURIER CALCULATIONS       */FFT 1040
   IF COPT= '3'                                                      FFT 1050
   THEN GO TO RETURN,.                /****************************/FFT 1060
REAL..                                /*REAL VALUES FROM (FOR)     */FFT 1070
   I    =1,.                          /*COMPLEX FOURIER TRANSFORM  */FFT 1080
        DO K =3 TO NH-1 BY 2,.        /****************************/FFT 1090
        J    =N-K+2,.                                                FFT 1100
        AAR  =A(K)  +A(J),                                           FFT 1110
        AAI  =A(K+1)-A(J+1),.                                        FFT 1120
        ABR  =A(K+1)+A(J+1),.                                        FFT 1130
        ABI  =A(J)  -A(K),.                                          FFT 1140
        I    =I+1,.                                                  FFT 1150
        SI   =S(I),.                  /*STORE SINE AND COSINE      */FFT 1160
        CO   =S(NQ-I),.                                              FFT 1170
        AW   = ABR*CO+ABI*SI,.                                       FFT 1180
        ABI  =-ABI*CO+ABR*SI,.                                       FFT 1190
        A(K) =( AAR+AW )*1E-1B,.                                     FFT 1200
```

```
        A(K+1)=(-AAI+ABI)*1E-1B,.                                    FFT 1210
        A(J) =( AAR-AW )*1E-1B,.                                     FFT 1220
        A(J+1)=( AAI+ABI)*1E-1B,.                                    FFT 1230
        END,.                                                        FFT 1240
   AW   =A(1),.                                                      FFT 1250
   IF COPT= '2'                       /*PREPARE  A(1),A(2)  FOR    */FFT 1260
   THEN DO,.                          /*CALCULATION OF REAL FOURIER */FFT 1270
        A(1) =(AW+A(N+1)),.           /*SERIES                     */FFT 1280
        A(2) =(AW-A(N+1)),.                                          FFT 1290
        COPT = '3',.                  /*CHANGE INTERNAL OPTION TERM */FFT 1300
        GO TO INV,.                                                  FFT 1310
        END,.                                                        FFT 1320
   A(1) =(AW+A(2))*1E-1B,.            /*CALCULATE VALUES           */FFT 1330
   A(N+1)=(AW-A(2))*1E-1B,.           /*A(1),A(2),A(N+1),A(N+2)    */FFT 1340
   A(2) =0,.                                                         FFT 1350
   A(N+2)=0,.                                                        FFT 1360
RETURN..                                                             FFT 1370
   END,.                              /*END OF PROCEDURE FFT       */FFT 1380
```

Purpose:

FFT performs finite one-dimensional Fourier analysis and synthesis for a set of $N=2^M$ real data, or for a sequence of $\frac{N}{2} = 2^{M-1}$ complex data.

Depending on the character of the input parameter OPT, the following transformations can be done:

OPT	=	'0'	real analysis
OPT	=	'1'	complex analysis
OPT	=	'2'	real synthesis
OPT	=	'3'	complex synthesis

Usage:

CALL FFT (A,M,OPT);

$A(2^M \text{ or } 2^M+2)$ - BINARY FLOAT [(53)]

Given one-dimensional array with length

$$\left\{\begin{matrix} N = 2^M \\ \\ N+2=2^M+2 \end{matrix}\right\} \text{for} \left\{\begin{matrix} \text{complex} \\ \\ \text{real} \end{matrix}\right\} \begin{matrix} \text{Fourier} \\ \text{calculations.} \end{matrix}$$

Resultant transform values are returned in the array A, replacing the input data.

The contents of the input and output array A depend on the option parameter OPT:

In cases OPT = '1' and OPT = '3' the complex data are located by pairs in N immediately adjacent storage locations. In the other cases the N function values are stored in N successive storage locations, while the Fourier coefficients a(n), b(n) need N+2 locations and they are stored as follows:

$$\frac{a_0}{2},\ b_0 = 0,\ a_1,\ b_1,\ a_2,\ b_2,$$

$$\ldots,\ a_{\frac{N}{2}-1},\ b_{\frac{N}{2}-1},\ \frac{a_{\frac{N}{2}}}{2},$$

$$b_{\frac{N}{2}} = 0$$

M - BINARY FIXED
Given integer that determines the size of vector A.
The size of A is

$$\left\{\begin{matrix} 2^M \\ \\ 2^M + 2 \end{matrix}\right\} \text{for} \left\{\begin{matrix} \text{complex} \\ \\ \text{real} \end{matrix}\right\} \text{Fourier calculations.}$$

OPT - CHARACTER(1)
Given option parameter for selection of operation (see "Purpose").

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='P' means error in specified parameter -- for example, $M < 2$. Any value of OPT different from '1', '2', '3' is treated as if it were '0'. The integer N in the given formulas (see "Purpose") must be a power of two:

$$N = 2^M$$

FFT is restricted to one-dimensional Fourier transformations.

Another procedure, called FFTM, is available in SSP-PL/I which operates on multidimensional arrays.

For real and complex applications of FFT the following is true: A forward transform (Fourier analysis) followed by an inverse transform (Fourier synthesis) returns the original data (except for roundoff errors).

Method:

Calculations depending on the option parameter OPT are done using the Cooley-Tukey Fast Fourier Transform.

For reference see:

J. W. Cooley, P. A. W. Lewis, P. D. Welch, "The Fast Fourier Transform Algorithm and its Applications", IBM Research, RC 1743, February 9, 1967, pp. 15-33.

N. M. Brenner, "Three Fortran Programs that Perform the Cooley-Tukey Fourier Transform", Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Technical Note ESD-TR-67-462, 1967.

J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computations, vol. 19, 1965, pp. 297-301.

Mathematical Background:

Complex Fourier calculations

Let X(k), k = 0, 1, 2,..., N-1, be a sequence of $N = 2^M$ complex numbers. The finite Fourier transform of X(k) is defined as

$$A(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot W_N^{-n \cdot k}$$

$$n = 0, 1, \ldots, N-1 \qquad (1)$$

where

$$W_N = \exp\left(\frac{2\pi i}{N}\right) \text{ and } i = \sqrt{-1}$$

Similarly, X(k) can be expressed as the finite Fourier series of A(n)

$$X(k) = \sum_{n=0}^{N-1} A(n) \cdot W_N^{n \cdot k} \qquad (2)$$

Since $N = 2^M$ we express X(k) as a function of the M arguments $k_{M-1}, k_{M-2} \cdots k_1, k_0$ of the binary representation of k:

$$k = k_{M-1} \cdot 2^{M-1} + k_{M-2} \cdot 2^{M-2}$$

$$+ \ldots k_1 \cdot 2 + k_0; k_v = 0 \text{ or } 1. \qquad (3)$$

Analogously, if

$$n = n_{M-1} \cdot 2^{M-1} + n_{M-2} \cdot 2^{M-2} + \ldots n_1 \cdot 2$$

$$+ n_0 ; n_v = 0 \text{ or } 1, \qquad (4)$$

then equation (2) can be written:

$$X(k_{M-1}, k_{M-2}, \ldots, k_1, k_0)$$

$$= \sum_{n_0=0}^{1} \sum_{n_1=0}^{1} \cdots \sum_{n_{M-1}=0}^{1} A(n_{M-1}, n_{M-2},$$

$$\ldots n_1, n_0) \cdot W_N^{k(n_{M-1} \cdot 2^{M-1}}$$

$$+ \ldots + n_1 \cdot 2 + n_0) \qquad (5)$$

Using $W_N^{2^M} = W_N^N = 1$, we have

$$W_N^{k \cdot n_{M-1} \cdot 2^{M-1}} = W_N^{k_0 \cdot n_{M-1} \cdot 2^{M-1}}$$

Therefore the innermost sum in equation (5) yields an array:

$$A_1 (k_0, n_{M-2}, \ldots, n_1, n_0) = \sum_{n_{M-1}=0}^{1} A \; (n_{M-1}, n_{M-2}, \ldots, n_1, n_0) \cdot W_N^{k_0 n_{M-1} \cdot 2^{M-1}}$$

Then, summing over n_{M-2} to get an array A_2 from A_1, and so on, leads to the general formula (L = 1, 2, 3, ..., M):

$$A_L (k_0, \ldots, k_{L-1}, n_{M-L-1}, \ldots, n_1, n_0)$$

$$= \sum_{n_{M-L}=0}^{1} A_{L-1} (k_0, \ldots, k_{L-2}, n_{M-L}, n_{M-L-1}, \ldots, n_1, n_0) \cdot W_N^{(k_{L-1} \cdot 2^{L-1} + \ldots + k_0) \cdot n_{M-L} \cdot 2^{M-L}}$$

The final array will be the desired X. The storage indexing convention used here is to let the M arguments of $A_L (k_0, \ldots, n_0)$ be the binary representation of the index of the storage location for $A_L (k_0, \ldots, n_0)$. In this way, each step of the algorithm involves fetching from two storage locations and returning results in the same two locations, thereby saving storage. However, the elements of the final array are in wrong order:

$$X (k_{M-1}, k_{M-2}, \ldots, k_1, k_0) = A_M(k_0, k_1, \ldots, k_{M-1})$$

Now we must reverse the order of the bits in the binary representation of k. FFT does the reordering on the initial array so that the result is in the correct order.

Real Fourier calculations

Given 2N real data Y(j), j = 0, 1, 2, ..., 2N-1. The coefficients of the trigonometric series

$$Y(j) = \frac{a(0)}{2} + \sum_{n=1}^{N-1} (a(n) \circ \cos \frac{\pi n j}{N} + b(n) \circ \sin \frac{\pi n j}{N}) + (-1)^j \frac{a(N)}{2}$$

can be derived from the N-point complex Fourier transform

$$A(n) = \frac{1}{N} \sum_{K=0}^{N-1} X(k) \circ W_N^{-n \cdot k} \qquad n = 0, 1, 2, \ldots, N-1$$

where X(k) = Y(2k) + iY(2k+1) ; k = 0, 1, 2, ..., N-1.

Let (the bar is conjugation):

$$2C(0) = \mathrm{Re}\, A(0) + \mathrm{Im}\, A(0)$$
$$2C(N) = \mathrm{Re}\, A(0) - \mathrm{Im}\, A(0)$$
$$2C(\tfrac{N}{2}) = \bar{A}\,(\tfrac{N}{2})$$

Calculate for $m = 1, 2, \ldots, \frac{N}{2} - 1$:

$$A_1 (m) = \frac{1}{2} (A(m) + \bar{A} (N-m))$$
$$\bar{A}_2 (N-m) = \frac{1}{2i} (A(m) - \bar{A}(N-m))$$
$$2C(m) = A_1 (m) + \bar{A}_2 (N-m) \circ W_{2N}^{-m}$$
$$2C(N-m) = A_1(m) - \bar{A}_2 (N-m) \cdot W_{2N}^{-m}$$

Now, identify the a(n), b(n) coefficients by means of the relations

$$a(0) = 2C(0)$$
$$a(N) = 2C(N)$$
$$\left.\begin{aligned} a(n) &= 2\mathrm{Re}C(n) \\ b(n) &= -2\,\mathrm{Im}C(n) \end{aligned}\right\} \; n = 1, 2, \ldots, N-1.$$

Note: To compute the 2N real Y(j) (Fourier synthesis) when the coefficients a(n) and b(n) are given, the process described above is applied in reverse order.

Programming Considerations:

FFT accepts input data stored according to option parameter OPT:

OPT = '1' } any set of $\frac{N}{2} = 2^{M-1}$ complex values
OPT = '3' } whose real and imaginary parts are located by pairs in N adjacent storage locations.

OPT = '2' the coefficients

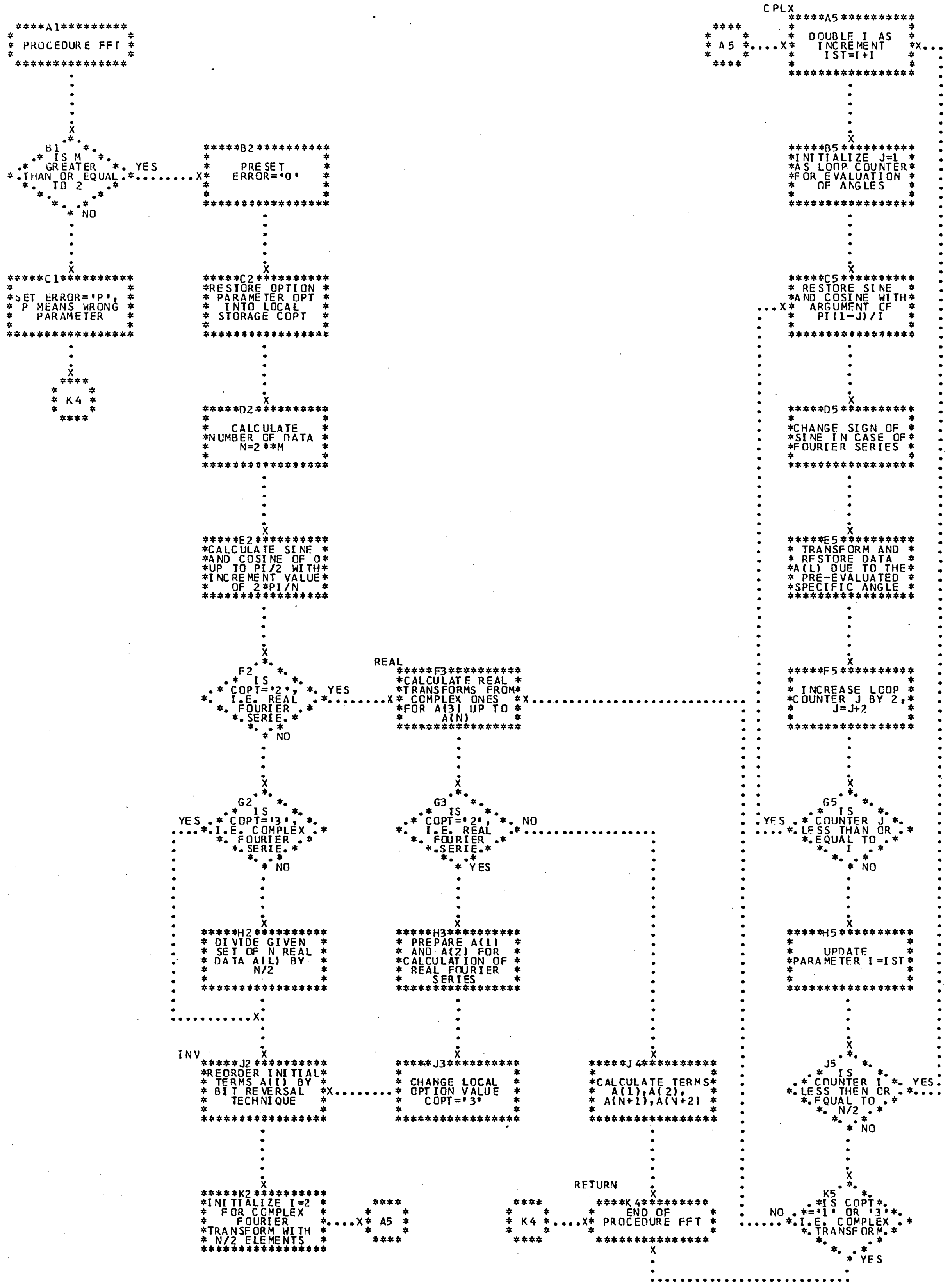
$$\frac{a_0}{2},\ b_0 = 0,\ a_1, b_1, \ldots,\ a_{\frac{N}{2}-1},\ b_{\frac{N}{2}-1},\ \frac{a_{\frac{N}{2}}}{2},\ b_{\frac{N}{2}} = 0$$

in N + 2 successive storage locations.

OPT = '0' N real elements in successive storage locations.

During calculation, input vector A is replaced by results depending on the character of parameter OPT. These results are stored in an analogous manner. For example, with OPT = '0', FFT calculates the N+2 Fourier coefficients a(n), b(n) and stores them into array A (with length N+2), overwriting the first N given real values.

PROCEDURE FFT PERFORMS FINITE, ONE-DIMENSIONAL FOURIER CALCULATIONS FOR A SET OF N=2**M REAL DATA A(L), L=1,2,...,N


A1
PROCEDURE FFT
B1
IS M GREATER THAN OR EQUAL TO 2
YES
NO
B2
PRESET ERROR='0'
C1
SET ERROR='P', P MEANS WRONG PARAMETER
K4
C2
RESTORE OPTION PARAMETER OPT INTO LOCAL STORAGE COPT
D2
CALCULATE NUMBER OF DATA N=2**M
E2
CALCULATE SINE AND COSINE OF 0 UP TO PI/2 WITH INCREMENT VALUE OF 2*PI/N
F2
IS COPT='2', I.E. REAL FOURIER SERIE
YES
NO
REAL
F3
CALCULATE REAL TRANSFORMS FROM COMPLEX ONES FOR A(3) UP TO A(N)
G2
IS COPT='3', I.E. COMPLEX FOURIER SERIE
YES
NO
G3
IS COPT='2', I.E. REAL FOURIER SERIE
NO
YES
H2
DIVIDE GIVEN SET OF N REAL DATA A(L) BY N/2
H3
PREPARE A(1) AND A(2) FOR CALCULATION OF REAL FOURIER SERIES
INV
J2
REORDER INITIAL TERMS A(I) BY BIT REVERSAL TECHNIQUE
J3
CHANGE LOCAL OPTION VALUE COPT='3'
J4
CALCULATE TERMS A(1),A(2), A(N+1),A(N+2)
K2
INITIALIZE I=2 FOR COMPLEX FOURIER TRANSFORM WITH N/2 ELEMENTS
A5
RETURN
K4
K4
END OF PROCEDURE FFT
CPLX
A5
A5
DOUBLE I AS INCREMENT IST=I+I
B5
INITIALIZE J=1 AS LOOP COUNTER FOR EVALUATION OF ANGLES
C5
RESTORE SINE AND COSINE WITH ARGUMENT OF PI(1-J)/I
D5
CHANGE SIGN OF SINE IN CASE OF FOURIER SERIES
E5
TRANSFORM AND RESTORE DATA A(L) DUE TO THE PRE-EVALUATED SPECIFIC ANGLE
F5
INCREASE LOOP COUNTER J BY 2, J=J+2
G5
IS COUNTER J LESS THAN OR EQUAL TO I
YES
NO
H5
UPDATE PARAMETER I=IST
J5
IS COUNTER I LESS THEN OR EQUAL TO N/2
YES
NO
K5
IS COPT='1' OR '3' I.E. COMPLEX TRANSFORM
NO
YES

Subroutine FFTM

```
FFTM..
/*********************************************************************/
/*                                                                   */
/*      FAST  FOURIER  TRANSFORM  FOR  MULTI-DIMENSIONAL  ARRAY      */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(A,M,NDIM,OPT),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR     */
        OPT CHARACTER(1),
        (A(*),PI,RI,RTH,TR,T2R,T2I,
        T3R,T3I,T4R,T4I,U1R,U1I,U2R,
        U2I,U3R,U3I,U4R,U4I,WR,WI,
        W2R,W2I,W3R,W3I)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (I,IND,J,JM,K,K2,K3,K4,KDIF,
        KINC,KM,KMIN,L,LJ,LMAX,M(*),
        MM,MMAX,N(NDIM),NA,NAD,NB,
        NBH,NDIM,NIN,NT)
        BINARY FIXED,.
   ERROR='P',.                        /*P MEANS WRONG PARAMETER      */
   IF NDIM LT 1                       /*TEST NUMBER OF DIMENSIONS    */
   THEN GO TO RETURN,.
   NT   =2,.
        DO I =1 TO NDIM,.
        N(I),K=10B**M(I),.            /*COMPUTE AND TEST DIMENSION   */
        IF K LT 1
        THEN GO TO RETURN,.           /*CALCULATE TOTAL NUMBER OF    */
        NT   =NT*K,.                  /*ELEMENTS                     */
        END,.
                                      /*COMPUTE  PI  AND  RTH        */
   PI   =3.141592653589793E+00,.
   RTH  =7.071067811865475E-01,.      /*RTH MEANS  SQRT(2)/2         */
   NA   =2,.                          /*******************************/
        DO IND =NDIM TO 1 BY -1,.     /*LOOP FOR EACH DIMENSION      */
        NIN  =N(IND),                 /*******************************/
        NB   =NA*NIN,.
        IF NIN= 1
        THEN GO TO MULTI,.
        NBH  =NB/10B,.
        J    =1,.                     /*******************************/
             DO I =1 TO NB BY NA,.    /*BIT REVERSAL TECHNIQUE       */
             IF J LE I                /*******************************/
             THEN GO TO MODI,.
             KM   =I+NA-2,.
             JM   =J-I,.
                  DO K =I TO KM BY 2,.
                       DO L =K TO NT BY NB,.
                       LJ   =L+JM,.
                       WR   =A(L),.  /*INTERCHANGE A(L) WITH A(LJ)  */
                       WI   =A(L+1),./*AND  A(L+1)  WITH  A(LJ+1)   */
                       A(L) =A(LJ),.
                       A(L+1)=A(LJ+1),.
                       A(LJ)=WR,.
                       A(LJ+1)=WI,.
                       END,.
                  END,.
MODI..                                /*MODIFY PARAMETER  J  AND  K  */
             K    =NBH,.
                  DO WHILE (J GT K),.
                  J    =J-K,.
                  K    =K/10B,.
                  END,.
             J    =J+K,.              /*COMPUTE NEW BIT REVERSAL     */
             END,.
        NAD  =NA+NA,.
ODD..                                 /*TEST FOR ODD M(IND)          */
        IF NIN LT 2
        THEN GO TO LEN4,.             /*M(IND) IS EVEN,    NIN = 1   */
        IF NIN= 2
        THEN GO TO LEN2,.             /*M(IND) IS ODD,     NIN = 2   */
        NIN  =NIN/100B,.
        GO TO ODD,.                   /*******************************/
LEN2..                                /*TRANSFORM WITH LENGTH 2      */
             DO I =1 TO NA BY 2,.     /*******************************/
                  DO K =I TO NT BY NAD,.
                  L    =K+NA,.
                  WR   =A(L),.
                  WI   =A(L+1),.
                  A(L) =A(K)-WR,.     /*MODIFY AND RESTORE ELEMENTS  */
                  A(L+1)=A(K+1)-WI,.
                  A(K) =A(K)+WR,.
                  A(K+1)=A(K+1)+WI,.
                  END,.
             END,.                    /*******************************/
LEN4..                                /*FAST  FOURIER  TRANSFORMS    */
        MMAX =NA,.                    /*WITH  LENGTH  4              */
MAIN..                                /*******************************/
        IF MMAX GE NBH
        THEN GO TO MULTI,.
        MM   =MMAX+MMAX,.
        LMAX =MAX(NAD,MMAX/10B),.
             DO I =NA TO LMAX BY NAD,./*EXECUTE LOOP FOR CALCULATION */
             J    =I,.                /*OF ANGLES FOR SPECIFIC MMAX  */
             IF MMAX LE NA
             THEN GO TO INITL,.
             RI   =-PI*J/MM,.
             IF OPT='1'
             THEN RI   =-RI,.         /*CHANGE SIGN FOR CALCULATION  */
             WR   =COS(RI),.          /*OF FOURIER SERIES            */
             WI   =SIN(RI),.
DOUBLE..
             W2R  =WR*WR-WI*WI,.      /*COMPUTE COSINE AND SINE      */
             W2I  =WR*WI*10E+00B,.    /*FOR   2*RI   AND   3*RI      */
             W3R  =W2R*WR-W2I*WI,.
             W3I  =W2R*WI+W2I*WR,.
INITL..                               /*INITIALIZE L AS INDEX FOR    */
             L    =1,.                /*MULTIDIMENSIONAL CALCULATIONS*/
STRT..
             IF MMAX= NA              /*COMPUTE START VALUE KMIN FOR */
             THEN KMIN =L,.           /*TRANSFORMATION LOOP          */
             ELSE KMIN =L+NIN*J,.
             KDIF =NIN*MMAX,.
INCR..                                /*COMPUTE INCREMENT FOR THE    */
             KINC =KDIF*100B,.        /*TRANSFORMATION LOOP          */
                  DO K =KMIN TO NT BY KINC,.
                  K2   =K +KDIF,.
                  K3   =K2+KDIF,.     /*K,K2,K3,K4  ARE PARAMETERS   */
                  K4   =K3+KDIF,.     /*FOR OPERATION WITH LENGTH 4  */
                  IF MMAX= NA         /*WITHOUT MULTIPLICATIONS      */
                  THEN DO,.
                       U1R  =A(K)   +A(K2),.
                       U1I  =A(K+1) +A(K2+1),.
                       U2R  =A(K3)  +A(K4),.
                       U2I  =A(K3+1)+A(K4+1),.
                       U3R  =A(K)   -A(K2),.
                       U3I  =A(K+1) -A(K2+1),.
                       U4R  =A(K3+1)-A(K4+1),.
                       U4I  =A(K4)  -A(K3),.
                       END,.
                  ELSE DO,.
                       T2R  =W2R*A(K2)  -W2I*A(K2+1),.
                       T2I  =W2R*A(K2+1)+W2I*A(K2),.
                       T3R  =WR *A(K3)  -WI *A(K3+1),.
                       T3I  =WR *A(K3+1)+WI *A(K3),.
                       T4R  =W3R*A(K4)  -W3I*A(K4+1),.
                       T4I  =W3R*A(K4+1)+W3I*A(K4),.
                       U1R  =A(K)  +T2R,.
                       U1I  =A(K+1)+T2I,.
                       U2R  =T3R   +T4R,.
                       U2I  =T3I   +T4I,.
                       U3R  =A(K)  -T2R,.
                       U3I  =A(K+1)-T2I,.
                       U4R  =T3I   -T4I,.
                       U4I  =T4R   -T3R,.
                       END,.
                  IF OPT= '1'         /*IN CASE OF FOURIER SERIES    */
                  THEN DO,.
                       U4R  =-U4R,.
                       U4I  =-U4I,.
                       END,.
                  A(K) =U1R+U2R,.     /*COMPUTE AND STORE NEW VALUES */
                  A(K+1)=U1I+U2I,.
                  A(K2)=U3R+U4R,.
                  A(K2+1)=U3I+U4I,.
                  A(K3)=U1R-U2R,.
                  A(K3+1)=U1I-U2I,.
                  A(K4)=U3R-U4R,.
                  A(K4+1)=U3I-U4I,.
                  END,.
             KMIN =L+(KMIN-L)*100B,.  /*UPDATE KMIN, KDIF AND IF NEC-*/
             KDIF =KINC,.             /*ESSARY REPEAT TRANSFORMATION */
             IF KDIF LE NBH           /*LOOP IN ORDER TO GET FINAL   */
             THEN GO TO INCR,.        /*VALUES                       */
                                      /*                             */
             L    =L+2,.              /*MODIFY L AND -IF NECESSARY-  */
             IF L LT NA               /*START ANOTHER TRANSFORM      */
             THEN GO TO STRT,.
                                      /*                             */
             J    =J+LMAX,.           /*MODIFY J AND -IF NECESSARY-  */
             IF J LE MMAX             /*THE ANGLE                    */
             THEN DO,.
                  TR   =WR,.          /*IF Z = COS(RI) + I*SIN(RI)   */
                  WR   =(TR+WI)*RTH,. /*THEN Z IS SUBSTITUTED BY     */
                  WI   =(WI-TR)*RTH,. /*Z  = Z * EXP(-PI/4 * I)      */
                  IF OPT= '1'
                  THEN DO,.
                       TR   =WR,.     /*Z IS SUBSTITUTED BY          */
                       WR   =-WI,.    /*Z  = Z * EXP(+PI/4 * I)      */
                       WI   =TR,.
                       END,.
                  GO TO DOUBLE,.
                  END,.
             END,.
        NIN  =3-NIN,.                 /*UPDATE NIN AND DOUBLE MMAX   */
        MMAX =MM,.
        GO TO MAIN,.
MULTI..
        NA   =NB,.
        END,.
   ERROR='0',.                        /*SUCCESSFUL FOURIER TRANSFORM */
RETURN..
   END,.                              /*END OF PROCEDURE FFTM        */
```

Purpose:

FFTM performs finite, multidimensional Fourier forward or inverse transformations for complex arrays whose dimensions are powers of two.

Depending on the value of the input parameter OPT, the following transformations can be done:

OPT = '0'   forward Fourier transform
OPT = '1'   inverse Fourier transform

Usage:

CALL FFTM (A, M, NDIM, OPT);

$A\ (2^{1+M_1+M_2\ldots M_{NDIM}})$ -

BINARY FLOAT [(53)]
Given one-dimensional real array used to hold the complex multidimensional array $A(N_1, N_2, \ldots, N_{NDIM})$ to be transformed.
The real and the imaginary parts of a data element must be placed by pairs

into immediately adjacent locations in storage. Note that the last subscript increases most rapidly.
Resultant complex Fourier transform in the same storage order.
The number of elements of vector A is

$$2\cdot N_1\cdot N_2\cdot\ldots N_{NDIM} = 2^{1+M_1+M_2}\ldots M_{NDIM}$$

M(NDIM) - BINARY FIXED
Given integer vector of length NDIM, which determines the extent of each dimension of complex array $A(N_1, N_2, N_{NDIM})$:

$$N_1 = 2^{M(1)},\ N_2 = 2^{M(2)},\ \ldots\ N_{NDIM} = 2^{M(NDIM)}$$

NDIM - BINARY FIXED
Given number of dimensions of multidimensional array A.

OPT - CHARACTER (1)
Given option parameter for selection of transform.

Remarks:

Procedure FFTM is to be used for Fourier transforms of complex, multidimensional arrays in which each dimension is a power of two:

$$N_\nu = 2^{M(\nu)} \text{ with } \nu = 1,2,\ldots,NDIM$$

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero.

Error parameter ERROR='P' is returned if NDIM < 1 or any $N_\nu < 1$.

A forward transform followed by an inverse transform, returns the original data multiplied by $N_1 \cdot N_2\cdot\ldots N_{NDIM}$ (except for roundoff errors).

Method:

Calculations performed are based on the Cooley-Tukey Fast Fourier transform.

For reference see:

J. W. Cooley, P. A. W. Lewis, P. D. Welch, "The Fast Fourier Transform Algorithm and its Applications", IBM Research, RC 1743, February 9, 1967, pp. 15-30.

N. M. Bremner, "Three Fortran Programs that Perform the Cooley-Tukey Fourier Transform", Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Technical Note ESD-TR-67-462, 1967.

J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computations, vol. 19, 1965, pp. 297-301.

Mathematical Background

The normal algorithm

Let $B(n_1, n_2, \ldots, n_L)$ be a complex multidimensional array whose dimensions are powers of two:

$$N_\nu = 2^{M(\nu)},\ \nu = 1,2,\ldots,L$$

The finite Fourier forward transform of B is defined as

$$A(k_1,\ldots k_L) = \frac{1}{N_1\cdot N_2\cdot\ldots N_L}\sum_{n_1=0}^{N_1-1}\cdots\sum_{n_L=0}^{N_L-1} B(n_1,\ldots, n_L) \cdot W_1^{-n1\cdot k_1}\cdot\ldots\cdot W_L^{-n_L\cdot k_L} \quad (1)$$

where:

$$W_\nu = \exp\left(\frac{2\pi i}{N_\nu}\right) \text{and } I = \sqrt{-1}$$

Similarly, $B(n_1,\ldots,n_L)$ can be expressed as the finite Fourier inverse transform (or Fourier series) of $A(k_1,\ldots,k_L)$.

$$B(n_1,\ldots,n_L) = \sum_{k_1=0}^{N_1-1}\cdots\sum_{k_L=0}^{N_L-1} A(k_1,\ldots,k_L) \cdot W_1^{+k_1\cdot n_1}\cdot\ldots\cdot W_L^{+k_L\cdot n_L} \quad (2)$$

The innermost sum yields an array

$$A_1(k_1,\ldots,k_{L-1},n_L) = \sum_{k_L=0}^{N_L-1} A(k_1,\ldots,k_L)\cdot W_L^{+k_L\cdot n_L} \quad (3)$$

Since equation (3) is equivalent to a one-dimensional problem, we discuss now the algorithm for one-dimensional complex Fourier transform.

$$X(n) = \sum_{k=0}^{N-1} A(k) \cdot W_N^{k\cdot n} \;, \quad W_N = \exp\left(\frac{2\pi i}{N}\right) \qquad (4)$$

Since $N = 2^M$, we express $X(n)$ as a function of the M arguments $n_{M-1}, n_{M-2}, \ldots n_1, n_0$ of the binary representation of n:

$$n = n_{M-1} \cdot 2^{M-1} + n_{M-2} \cdot 2^{M-2} + \ldots + n_1 \cdot 2 + n_0 ; \; n_\nu = 0 \text{ or } 1.$$

Analogously, if

$$k = k_{M-1} \cdot 2^{M-1} + k_{M-2} \cdot 2^{M-2} + \ldots + k_1 \cdot 2 + k_0 \;\; ; \;\; k_\nu = 0 \text{ or } 1$$

then equation (4) can be written:

$$X(n_{M-1}, n_{M-2}, \ldots n_1, n_0) = \sum_{k_0=0}^{1} \cdots \sum_{k_{M-1}=0}^{1} A(k_{M-1}, k_{M-2}, \ldots, k_1, k_0) \, W_N^{n\cdot(k_{M-1}\cdot 2^{M-1} + \ldots + k_1 \cdot 2 + k_0)} \qquad (5)$$

Using $W_N^{2M} = W_N^{N} = 1$, we have

$$W_N^{n\cdot k_{M-1}\cdot 2^{M-1}} = W_N^{n_0 \cdot k_{M-1} \cdot 2^{M-1}}$$

Therefore the innermost sum in equation (5) yields an array:

$$A_1(n_0, k_{M-2}, \ldots, k_1, k_0) = \sum_{k_{M-1}=0}^{1} A(k_{M-1}, k_{M-2}, \ldots, k_1, k_0) \cdot W_N^{n_0 \cdot k_{M-1} \cdot 2^{M-1}}$$

Then, summation over k_{M-2}, to get an array A_2 from A_1, and so on, leads to the general formula $(L = 1, 2, \ldots, M)$:

$$A_L(n_0, \ldots n_{L-1}, k_{M-L-1}, \ldots k_0) = \sum_{k_{M-L}=0}^{1} A_{L-1}(n_0, \ldots, n_{L-2}, k_{M-L}, k_{M-L-1}, \ldots, k_1, k_0) \cdot W_N^{(n_{L-1}\cdot 2^{L-1} + \ldots + n_1 \cdot 2 + n_0) \cdot k_{M-L} \cdot 2^{M-L}} \qquad (6)$$

The final array will be the desired X. The storage indexing convention used here is to let the M arguments of $A_L(n_0, \ldots, k_0)$ be the binary representation of the index of the storage location for $A_L(n_0, \ldots, k_0)$. In this way, each step of the algorithm involves fetching from two storage locations and returning results in the same two locations, thereby saving storage. However, the elements of the final array are in wrong order:

$$X(n_{M-1}, n_{M-2}, \ldots, n_1, n_0) = A_M(n_0, n_1, \ldots, n_{M-1})$$

Now, we must reverse the order of the bits in the binary representation of n. FFT does the reordering on the initial array so that the result is in the correct order.

The two-step algorithm

A modification that achieves further economy at the expense of program complexity is to take two steps at a time when the A_L in equation (6) are calculated. Let us define J as the index given by the high-order L-2 bit positions of an index and let K be the low-order M-L bit positions:

$$A_L(\underbrace{n_0, \ldots, n_{L-3}}_{J}, n_{L-2}, n_{L-1}, \underbrace{k_{M-L-1}, \ldots, k_0}_{K}).$$

Let:

$$U = W_N^{(n_{L-3} \cdot 2^{L-3} + \ldots + n_1 2 + n_0) \cdot 2^{M-L}}$$

Then the step from L-2 to L-1, with

$$W_N^{2^{M-1}} = W_N^{\frac{N}{2}} = -1$$

is:

$$A_{L-1}(J,0,k_{M-L},K) = A_{L-2}(J,0,k_{M-L},K) + A_{L-2}(J,1,k_{M-L},K) \cdot U^2$$

$$A_{L-1}(J,1,k_{M-L},K) = A_{L-2}(J,0,k_{M-L},K) - A_{L-2}(J,1,k_{M-L},K) \cdot U^2 \qquad (7)$$

for $k_{M-L} = 0, 1$.

For the step from L-1 to L, we make use of the fact that

$$W_N^{2^{M-2}} = W_N^{\frac{N}{4}} = i \text{ and get:}$$

$$A_L(J,n_{L-2},0,K) = A_{L-1}(J,n_{L-2},0,K) + A_{L-1}(J,n_{L-2},1,K) \cdot i^{n_{L-2}} \cdot U$$

$$A_L(J,n_{L-2},1,K) = A_{L-1}(J,n_{L-2},0,K) - A_{L-1}(J,n_{L-2},1,K) \cdot i^{n_{L-2}} \cdot U \qquad (8)$$

for $n_{L-2} = 0, 1$.

Dropping J and K to simplify notation, we write equations (7) and (8) in a form that requires only three instead of four complex multiplications. To do this, let

$$\overline{A}_{L-1}(n_{L-2},1) = A_{L-1}(J,n_{L-2},1,K) \cdot U$$

Then, we have:

<u>for $k_{M-L} = 0$</u>

$$A_{L-1}(0,0) = A_{L-2}(0,0) + A_{L-2}(1,0) \cdot U^2$$

$$A_{L-1}(1,0) = A_{L-2}(0,0) - A_{L-2}(1,0) \cdot U^2$$

<u>for $k_{M-L} = 1$</u>

$$\overline{A}_{L-1}(0,1) = A_{L-2}(0,1) \cdot U + A_{L-2}(1,1) \cdot U^3$$

$$\overline{A}_{L-1}(1,1) = A_{L-2}(0,1) \cdot U - A_{L-2}(1,1) \cdot U^3$$

<u>for $n_{L-2} = 0$</u> (9)

$$A_L(0,0) = A_{L-1}(0,0) + \overline{A}_{L-1}(0,1)$$

$$A_L(0,1) = A_{L-1}(0,0) - \overline{A}_{L-1}(0,1)$$

<u>for $n_{L-2} = 1$</u>

$$A_L(1,0) = A_{L-1}(1,0) + \overline{A}_{L-1}(1,1) \cdot i$$

$$A_L(1,1) = A_{L-1}(1,0) - \overline{A}_{L-1}(1,1) \cdot i$$

These equations are used for L = 2, 4, 6, . . . , M, if M is even. If M is odd, a single step is taken with L = 1 and equations (9) are used with L = 3, 5, 7, . . . , M.

The cases with J = 0 and J = 1 are programmed separately to avoid multiplications:

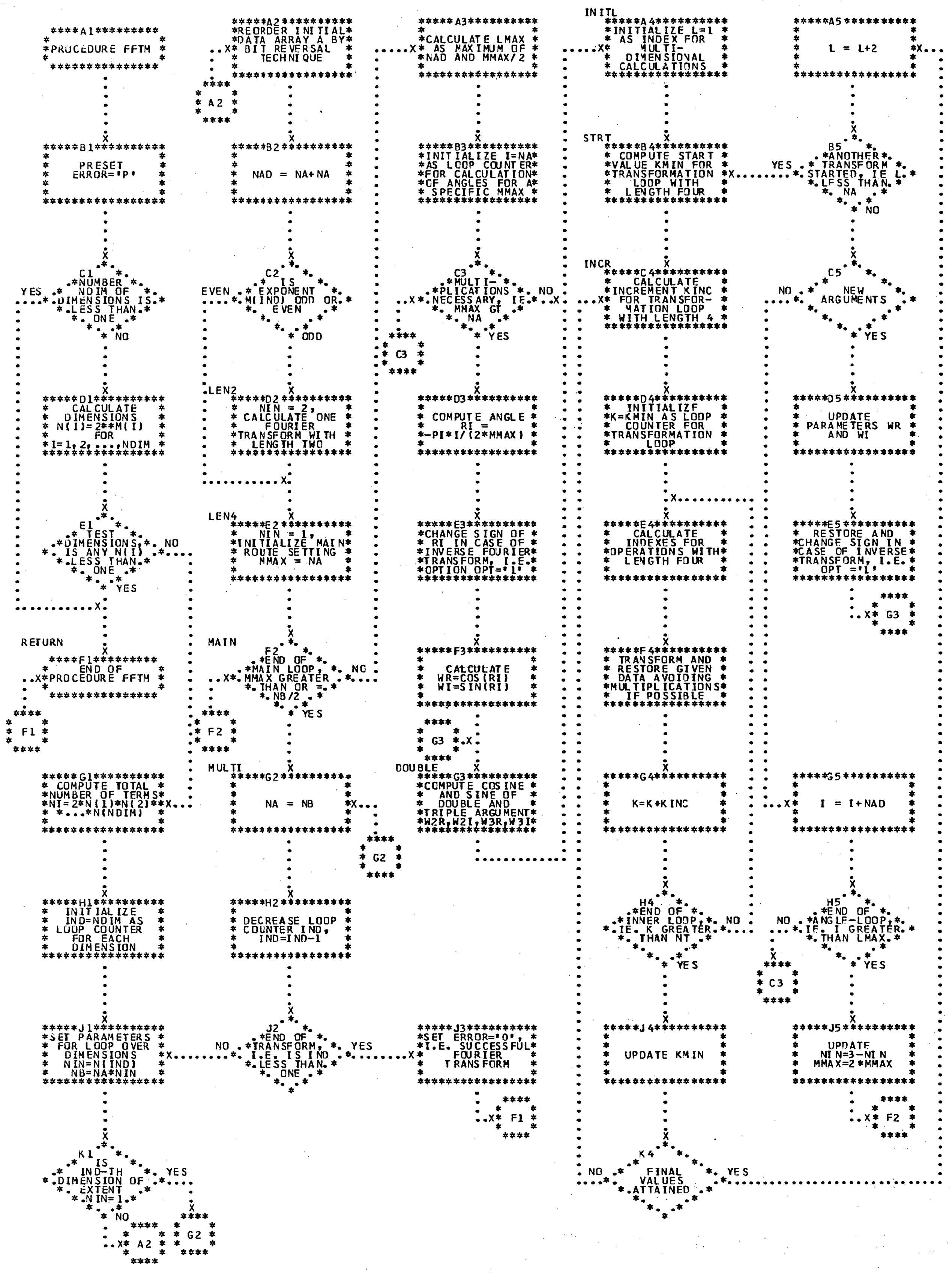
J = 0 gives U = 1

$$J = 1 \text{ gives } U = W_N^{2^{L-3} \cdot 2^{M-L}}$$

$$= W_N^{\frac{N}{8}} = e^{\frac{\pi i}{4}} = \frac{1}{\sqrt{2}}(1+i)$$

and $U^2 = i$, $U^3 = \frac{1}{\sqrt{2}}(i-1)$.

FFIM PERFORMS FINITE, MULTIDIMENSIONAL FOURIER TRANSFORMS FOR COMPLEX ARRAYS, WHOSE DIMENSIONS ARE POWERS OF TWO



A1
PROCEDURE FFTM
B1
PRESET ERROR='P'
C1
NUMBER NDIM OF DIMENSIONS IS LESS THAN ONE
YES
NO
D1
CALCULATE DIMENSIONS N(I)=2**M(I) FOR I=1, 2, ..., NDIM
E1
TEST DIMENSIONS, IS ANY N(I) LESS THAN ONE
NO
YES
RETURN
F1
END OF PROCEDURE FFTM
F1
G1
COMPUTE TOTAL NUMBER OF TERMS NT=2*N(1)*N(2)* *...*N(NDIM)
H1
INITIALIZE IND=NDIM AS LOOP COUNTER FOR EACH DIMENSION
J1
SET PARAMETERS FOR LOOP OVER DIMENSIONS NIN=N(IND) NB=NA*NIN
K1
IS IND-TH DIMENSION OF EXTENT NIN=1
YES
NO
A2
G2
A2
REORDER INITIAL DATA ARRAY A BY BIT REVERSAL TECHNIQUE
A2
B2
NAD = NA+NA
C2
IS EXPONENT M(IND) ODD OR EVEN
EVEN
ODD
LEN2
D2
NIN = 2, CALCULATE ONE FOURIER TRANSFORM WITH LENGTH TWO
LEN4
E2
NIN = 1, INITIALIZE MAIN ROUTE SETTING MMAX = NA
MAIN
F2
END OF MAIN LOOP, MMAX GREATER THAN OR = NB/2
NO
YES
F2
MULTI
G2
NA = NB
G2
H2
DECREASE LOOP COUNTER IND, IND=IND-1
J2
END OF TRANSFORM, I.E. IS IND LESS THAN ONE
NO
YES
A3
CALCULATE LMAX AS MAXIMUM OF NAD AND MMAX/2
B3
INITIALIZE I=NA AS LOOP COUNTER FOR CALCULATION OF ANGLES FOR A SPECIFIC MMAX
C3
MULTI-PLICATIONS NECESSARY, IE MMAX GT NA
NO
YES
C3
D3
COMPUTE ANGLE RI = -PI*I/(2*MMAX)
E3
CHANGE SIGN OF RI IN CASE OF INVERSE FOURIER TRANSFORM, I.E. OPTION OPT='1'
F3
CALCULATE WR=COS(RI) WI=SIN(RI)
G3
DOUBLE
G3
COMPUTE COSINE AND SINE OF DOUBLE AND TRIPLE ARGUMENT W2R,W2I,W3R,W3I
J3
SET ERROR='0', I.E. SUCCESSFUL FOURIER TRANSFORM
F1
INITL
A4
INITIALIZE L=1 AS INDEX FOR MULTI-DIMENSIONAL CALCULATIONS
STRT
B4
COMPUTE START VALUE KMIN FOR TRANSFORMATION LOOP WITH LENGTH FOUR
INCR
C4
CALCULATE INCREMENT KINC FOR TRANSFOR-MATION LOOP WITH LENGTH 4
D4
INITIALIZF K=KMIN AS LOOP COUNTER FOR TRANSFORMATION LOOP
E4
CALCULATE INDEXES FOR OPERATIONS WITH LENGTH FOUR
F4
TRANSFORM AND RESTORE GIVEN DATA AVOIDING MULTIPLICATIONS IF POSSIBLE
G4
K=K+KINC
H4
END OF INNER LOOP, IE. K GREATER THAN NT
NO
YES
J4
UPDATE KMIN
K4
FINAL VALUES ATTAINED
NO
YES
A5
L = L+2
B5
ANOTHER TRANSFORM STARTED, IE L LFSS THAN NA
YES
NO
C5
NEW ARGUMENTS
NO
YES
D5
UPDATE PARAMETERS WR AND WI
E5
RESTORE AND CHANGE SIGN IN CASE OF INVERSE TRANSFORM, I.E. OPT ='1'
G3
G5
I = I+NAD
H5
END OF ANGLF-LOOP, IF. I GREATER THAN LMAX
NO
YES
C3
J5
UPDATE NIN=3-NIN MMAX=2*MMAX
F2

⊙ Subroutine APLL

```
APLL..
/*********************************************************************/
/*                                                                   */
/*      SET UP NORMAL EQUATIONS FOR A LINEAR LEAST SQUARES           */
/*      FIT TO A GIVEN DISCRETE FUNCTION                             */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(FCT,N,IP,WORK),.
   DECLAPE
      FCT ENTRY,
      (WORK(*),P(IP+1),A,WGT)
      BINARY FLOAT,               /*SINGLE PRECISION VERSION  /*S*/
 /*   BINARY FLOAT(53),           /*DOUBLE PRECISION VERSION  /*D*/
      (N,IP,LIP,IP1,I,J,K,L,M)
      BINARY FIXED,
      ERROR EXTERNAL CHARACTER(1),.
   ERROR='0',.                    /*SUCCESSFUL OPERATION          */
   LIP  =IP,.
   IP1  =LIP+1,.
   M    =IP1*(IP1+1)/2,.
      DO I =1 TO M,.              /*INIT. RIGHT HAND SIDE AND     */
      WORK(I)=0,.                 /*COEFFICIENT MATRIX EQUAL ZERO*/
      END,.
   IF N GT 0                      /*TEST SPECIFIED DIMENSIONS     */
   THEN IF LIP GT 0
   THEN IF N GT LIP
   THEN DO I =1 TO N,.            /*FOR I-TH ARGUMENT             */
                                  /*PROVIDE VALUES OF GIVEN FCT.,*/
      CALL FCT(I,N,LIP,P,WGT),.   /*WEIGHT AND FUNDAMENTAL FCT.   */
      IF ERROR NE '0'
      THEN GO TO OUT,.            /*ERROR IN PROCEDURE FCT.       */
      J    =0,.
         DO K =1 TO IP1,.         /*COMPUTE COEFFICIENT MATRIX    */
         A    =P(K)*WGT,.         /*AND RIGHT HAND SIDE           */
            DO L =1 TO K,.
            J    =J+1,.
            WORK(J)=WORK(J)+P(L)*A,.
            END,.
         END,.
      END,.
   ELSE ERROR='D',.               /*ERROR IN SPECIFIED DIMENSIONS*/
OUT..
   END,.                          /*END OF PROCEDURE APLL         */
```

Purpose:

APLL sets up the normal equations for a polynomial least squares fit to a given discrete function.

Usage:

CALL APLL (FCT, N, IP, WORK);

FCT - ENTRY
Given procedure supplying the values of the fundamental functions, of the function that is to be approximated and of the weights.
Usage:
CALL FCT (I, N, IP, P, WGT);

- I - BINARY FIXED
  Given subscript value for current point.
- N - BINARY FIXED
  Given number of points.
- IP - BINARY FIXED
  Given number of fundamental functions.
- P(IP+1) - BINARY FLOAT [(53)]
  Resultant vector containing values of fundamental functions, one up to IP, followed by value of function that must be approximated for the i-th argument.
- WGT - BINARY FLOAT [(53)]
  Resultant weight value for i-th argument.

N - BINARY FIXED
Given number of points.

IP - BINARY FIXED
Given number of fundamental functions.

WORK((IP+1)(IP+2)/2) -
BINARY FLOAT [(53)]
Resultant vector containing the lower triangular part of symmetric coefficient matrix of normal equations, stored rowwise, followed by right-hand side and square sum of function values.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='D' means error in specified dimensions IP, N -- that is, IP is not less than N or N not greater than 1.

For solving the normal equations, ASN may be used.

If ERROR is set to a nonzero value within procedure FCT, control is returned to the calling program.

Method:

The normal equations stored in the vector WORK are obtained by minimizing

$$\sum_{k=1}^{N} w(X_k) \left[f(X_k) - p(X_k) \right]^2$$

where:

$w(X_k)$ is the weight value for argument X_k
$f(X_k)$ is the value of the function to be approximated
$p(X_k)$ is the value of the approximation function

Mathematical Background:

Let $f(x)$, $g_i(x)$, $i = 1, 2, \ldots, IP$, and $w(x) > 0$ be functions defined for $x = x_1, x_2, \ldots, x_N$ (the x_i may be vectors as well as scalars).

The problem is to determine the coefficients c_i of the linear combination $p(x) \sum_{i=1}^{IP} c_i g_i(x)$ such that

$$\sum_{k=1}^{N} w(x_k) \left(f(x_k) - p(x_k)\right)^2 = \min.$$

This problem leads to a system of linear equations AC = R, where C is the vector of unknown coefficients, A is the IP by IP symmetric positive definite matrix with elements

$$a_{jk} = \sum_{i=1}^{N} w(x_i)\, g_j\,(x_i)\, g_k\,(x_i)$$

and R is an IP dimensional vector with elements

$$r_j = \sum_{i=1}^{N} w(x_i) f(x_i) g_j(x_i)$$

(See ASN for details.)

Some remarks regarding polynomial approximation are in order. Use of monomials $g_i(x) = x^{i-1}$ as fundamental functions results in a very badly conditioned coefficient matrix A. If Chebyshev or Legendre polynomials are used instead, the condition of the normal equations is improved remarkably, provided the arguments have a sensible distribution (for example, equidistant in the interval -1 to +1).

Programming Considerations:

To allow for full flexibility in data handling, the user must provide a procedure, described under "Usage".

Coefficient matrix A and right-hand side R are stored adjacently. Within a linear array WORK, the lower triangular part of A is stored rowwise, followed by R, which is augmented by one element, ff, in which the weighted square sum of function values is returned.

WORK = (a_{11}, a_{12}, a_{22}, ..., a_{1IP}, ..., $a_{IP\,IP}$, r_1, ..., r_{IP}, ff) represents a triangular array.

The described storage allocation of WORK is required by procedure ASN. The user has full flexibility for handling of the data

$$x_i,\ f(x_i),\ w\,(x_i),\ g_1\,(x_i), \ldots, g_{IP}\,(x_i)$$

1. If he wishes to allocate

$$x_i,\ f(x_i),\ w\,(x_i),\ g_1\,(x_i), \ldots,\ g_{IP}\,(x_i)$$

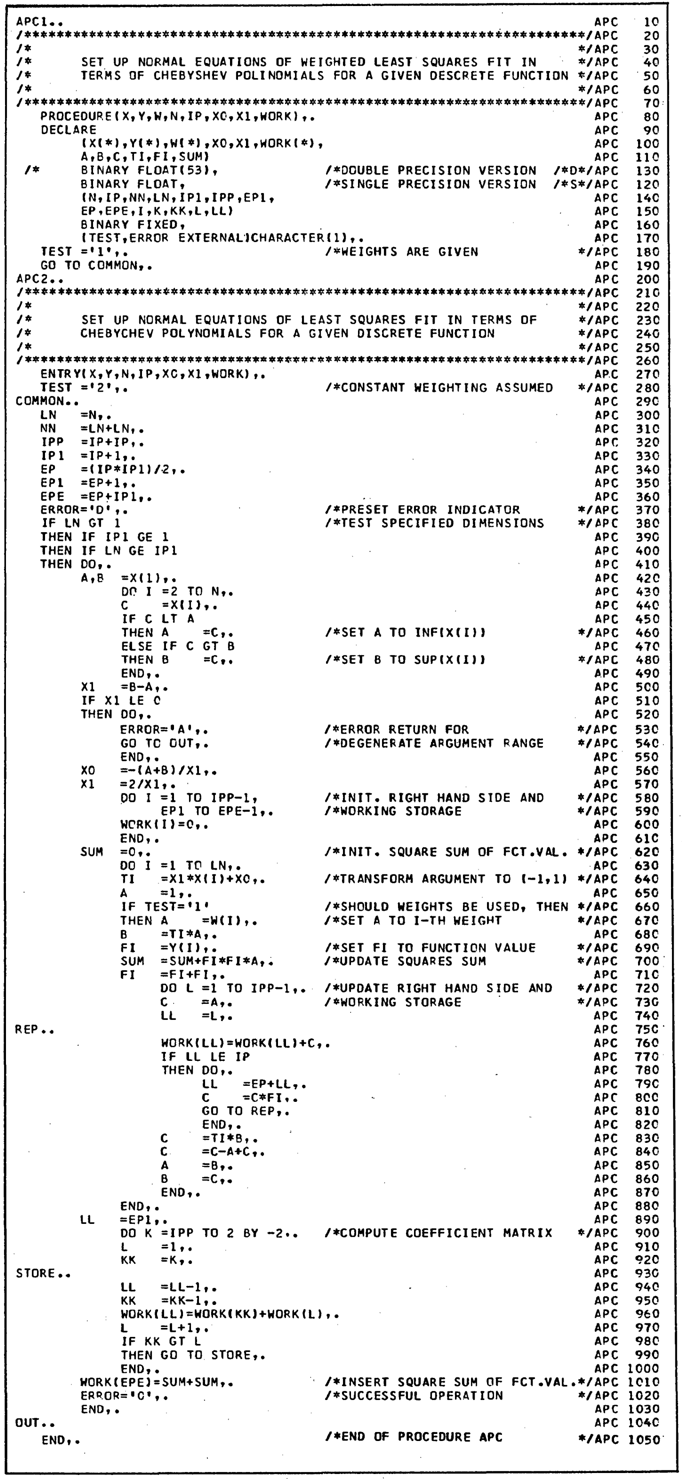
in main storage he may use external declarations.

2. Calculation of some or all of the required quantities as functions of the subscript or as functions of the argument x_i is another convenient choice.

3. The needed data may be read in sequentially from one or more external devices.

The three cases listed above may occur in any sensible combination.

- Subroutine APC1/APC2



```
APC1..
/*********************************************************************/
/*                                                                   */
/*      SET UP NORMAL EQUATIONS OF WEIGHTED LEAST SQUARES FIT IN     */
/*      TERMS OF CHEBYSHEV POLINOMIALS FOR A GIVEN DESCRETE FUNCTION */
/*                                                                   */
/*********************************************************************/
   PROCEDURE(X,Y,W,N,IP,XO,X1,WORK),.
   DECLARE
        (X(*),Y(*),W(*),XO,X1,WORK(*),
        A,B,C,TI,FI,SUM)
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
        (N,IP,NN,LN,IP1,IPP,EP1,
        EP,EPE,I,K,KK,L,LL)
        BINARY FIXED,
        (TEST,ERROR EXTERNAL)CHARACTER(1),.
   TEST ='1',.                        /*WEIGHTS ARE GIVEN            */
   GO TO COMMON,.
APC2..
/*********************************************************************/
/*                                                                   */
/*      SET UP NORMAL EQUATIONS OF LEAST SQUARES FIT IN TERMS OF     */
/*      CHEBYCHEV POLYNOMIALS FOR A GIVEN DISCRETE FUNCTION          */
/*                                                                   */
/*********************************************************************/
   ENTRY(X,Y,N,IP,XO,X1,WORK),.
   TEST ='2',.                        /*CONSTANT WEIGHTING ASSUMED   */
COMMON..
   LN   =N,.
   NN   =LN+LN,.
   IPP  =IP+IP,.
   IP1  =IP+1,.
   EP   =(IP*IP1)/2,.
   EP1  =EP+1,.
   EPE  =EP+IP1,.
   ERROR='0',.                        /*PRESET ERROR INDICATOR       */
   IF LN GT 1                         /*TEST SPECIFIED DIMENSIONS    */
   THEN IF IP1 GE 1
   THEN IF LN GE IP1
   THEN DO,.
        A,B  =X(1),.
             DO I =2 TO N,.
             C    =X(I),.
             IF C LT A
             THEN A    =C,.           /*SET A TO INF(X(I))           */
             ELSE IF C GT B
             THEN B    =C,.           /*SET B TO SUP(X(I))           */
             END,.
        X1   =B-A,.
        IF X1 LE 0
        THEN DO,.
             ERROR='A',.              /*ERROR RETURN FOR             */
             GO TO OUT,.              /*DEGENERATE ARGUMENT RANGE    */
             END,.
        XO   =-(A+B)/X1,.
        X1   =2/X1,.
             DO I =1 TO IPP-1,        /*INIT. RIGHT HAND SIDE AND    */
                  EP1 TO EPE-1,.      /*WORKING STORAGE              */
             WORK(I)=0,.
             END,.
        SUM  =0,.                     /*INIT. SQUARE SUM OF FCT.VAL. */
             DO I =1 TO LN,.
             TI   =X1*X(I)+XO,.       /*TRANSFORM ARGUMENT TO (-1,1) */
             A    =1,.
             IF TEST='1'              /*SHOULD WEIGHTS BE USED, THEN */
             THEN A    =W(I),.        /*SET A TO I-TH WEIGHT         */
             B    =TI*A,.
             FI   =Y(I),.             /*SET FI TO FUNCTION VALUE     */
             SUM  =SUM+FI*FI*A,.      /*UPDATE SQUARES SUM           */
             FI   =FI+FI,.
                  DO L =1 TO IPP-1,.  /*UPDATE RIGHT HAND SIDE AND   */
                  C    =A,.           /*WORKING STORAGE              */
                  LL   =L,.
REP..
                  WORK(LL)=WORK(LL)+C,.
                  IF LL LE IP
                  THEN DO,.
                       LL   =EP+LL,.
                       C    =C*FI,.
                       GO TO REP,.
                       END,.
                  C    =TI*B,.
                  C    =C-A+C,.
                  A    =B,.
                  B    =C,.
                  END,.
             END,.
        LL   =EP1,.
             DO K =IPP TO 2 BY -2..   /*COMPUTE COEFFICIENT MATRIX   */
             L    =1,.
             KK   =K,.
STORE..
             LL   =LL-1,.
             KK   =KK-1,.
             WORK(LL)=WORK(KK)+WORK(L),.
             L    =L+1,.
             IF KK GT L
             THEN GO TO STORE,.
             END,.
        WORK(EPE)=SUM+SUM,.           /*INSERT SQUARE SUM OF FCT.VAL.*/
        ERROR='0',.                   /*SUCCESSFUL OPERATION         */
        END,.
OUT..
   END,.                              /*END OF PROCEDURE APC         */
```

Purpose:

APC1/APC2 sets up the normal equations for a polynomial least squares fit to a given discrete function, using Chebyshev polynomials as fundamental functions.

Usage:

CALL APC1 (X, Y, W, N, IP, X0, X1, WORK);

X(N) - BINARY FLOAT [(53)]
Given vector of argument values.

Y(N) - BINARY FLOAT [(53)]
Given vector of function values that are to be approximated.

W(N) - BINARY FLOAT [(53)]
Given vector of weighted values.

N - BINARY FIXED
Given number of argument values.

IP - BINARY FIXED
Given number of Chebyshev polynomials.

X0 - BINARY FLOAT [(53)]
Resultant additive constant for linear transformation of argument range.

X1 - BINARY FLOAT [(53)]
Resultant multiplicative constant for linear transformation of argument range.

WORK((IP+1)(IP+2)/2) -
BINARY FLOAT [(53)]
Resultant vector containing the lower triangular part of symmetric coefficient matrix of normal equations, stored rowwise, followed by right-hand side and square sum of function values.

CALL APC2 (X, Y, N, IP, X0, X1, WORK);

X(N) - BINARY FLOAT [(53)]
Given vector of argument values.

Y(N) - BINARY FLOAT [(53)]
Given vector of function values that are to be approximated.

N - BINARY FIXED
Given number of argument values.

IP - BINARY FIXED
Given number of Chebyshev polynomials.

X0 - BINARY FLOAT [(53)]
Resultant additive constant for linear transformation of argument range.

X1 - BINARY FLOAT [(53)]
Resultant multiplicative constant for linear transformation of argument range.

WORK((IP+1)(IP+2)/2) -
BINARY FLOAT [(53)]
Resultant vector containing the lower triangular part of symmetric coefficient matrix of normal equations, stored rowwise, followed by right-hand side and square sum of function values.

Remarks:

1. If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='D' means error in specified dimensions IP, N -- that is, for IP not less than N or N not greater than 1.

2. APC2 implies constant weighting (value one).

3. The use of Chebyshev polynomials instead of monomials results in a remarkable improvement of the condition of the normal equations, provided the arguments have a sensible distribution (for example, equidistant).

4. The given argument range is reduced by means of the linear transformation.

$$t(x) = x_1 \cdot x + x_0$$

to the reduced range $-1 \le t(x) \le +1$. The normal equations are set up for Chebyshev expansions in t(x) and the solution of these equations is determined by procedure ASN. This is no disadvantage, since the Chebyshev expansion may be evaluated effectively for a specified argument x using procedure POSV with argument $t = x \cdot x_1 + x_0$ and the calculated coefficient vector of the Chebyshev expansion.

5. The transformation of the calculated Chebyshev expansion to an ordinary polynomial may be accomplished using procedure POST.

Method:

The polynomial fit is calculated in the form of its Chebyshev expansion.

$$C_1\, T_0\,(t) + C_2\, T_1\,(t) + \ldots + C_{IP}\, T_{IP-1}\,(t)$$

where $T_k\,(t)$ is the Chebyshev polynomial of degree k.

The values of the Chebyshev polynomials for the argument t are calculated by means of the three-term recurrence equation:

$$T_k\,(t) = 2\,t\,T_{k-1}\,(t) - T_{k-2}\,(t);\ k \ge 2$$

with starting values $T_0\,(t) = 1$, $T_1\,(t) = t$. In setting up the coefficient matrix, time is saved by using the identity

$$2\,T_j \cdot T_k = T_{j+k} + T_{|j-k|}$$

Mathematical Background:

Let x_L and x_R denote the leftmost and rightmost arguments respectively. By means of the linear transformation

$$t(x) = \frac{2x-(x_L + x_R)}{x_R - x_L} = x_1 \circ x + x_0$$

the argument range $x_L \le x \le x_R$ is reduced to the argument range $-1 \le t(x) \le +1$.

The function f(x), given for $x = x_1, x_2, \ldots, x_N$, is to be approximated by an expansion in Chebyshev polynomials:

$$p(x) = \sum_0^{IP} C_i \, T_{i-1} (t(x))$$

so that

$$\sum_{i=1}^{N} w(x_i) \, [f(x_i) - p(x_i)]^2 = \min.$$

T_k (t) is the Chebyshev polynomial of degree k.

The vector C of unknown coefficients C_i is a solution of the matrix equation AC = R, where A is an IP by IP symmetric positive definite matrix with elements

$$a_{jk} = \sum_{i=1}^{N} w(x_i) \, T_{j-1} (t(x_i)) \, T_{k-1} (t(x_i))$$

and R is a vector of dimension IP with elements

$$r_j = \sum_{i=1}^{N} w(x_i) \, T_{j-1} (t(x_i)) \, f(x_i)$$

(See ASN for details.)

The Chebyshev expansion of the polynomial p(x) gives a much better indication of the accuracy of the approximation than the coefficient vector of the polynomial itself. If the specified degree of the polynomial is too high, the last terms of the Chebyshev expansion are uniformly small compared with the preceding coefficients. The degree might be reduced by the number of small trailing coefficients without unduly enlarging the overall error.

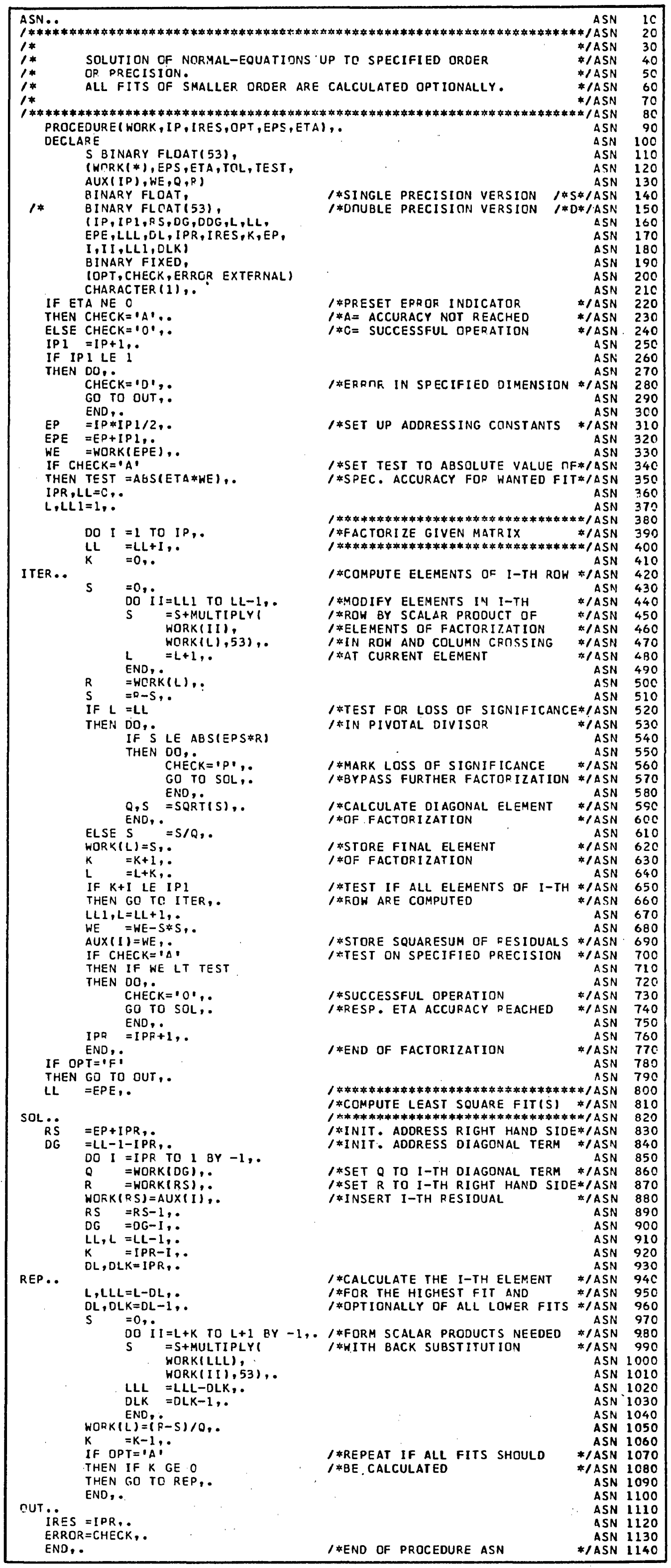
An upper bound for the error introduced by neglecting the last terms of the Chebyshev expansion is given by the sum of the absolute values of these terms. Normally, transformation of the Chebyshev expansion in t(x) to ordinary polynomials in x results in severe loss of accuracy. Therefore, no attempt is made to return the polynomial expansions.

Programming Considerations:

Only the lower triangular part of the symmetric coefficient matrix is generated and stored rowwise, followed immediately by the right-hand side and by the weighted square sum of function values.

This storage allocation scheme is required by subroutine ASN, which may be used for calculation of the normal equations.

• Subroutine ASN



```
ASN..
/******************************************************************/
/*                                                                */
/*      SOLUTION OF NORMAL-EQUATIONS UP TO SPECIFIED ORDER        */
/*      OR PRECISION.                                             */
/*      ALL FITS OF SMALLER ORDER ARE CALCULATED OPTIONALLY.      */
/*                                                                */
/******************************************************************/
   PROCEDURE(WORK,IP,IRES,OPT,EPS,ETA),.
   DECLARE
        S BINARY FLOAT(53),
        (WORK(*),EPS,ETA,TOL,TEST,
        AUX(IP),WE,Q,R)
        BINARY FLOAT,              /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),          /*DOUBLE PRECISION VERSION  /*D*/
        (IP,IP1,RS,DG,DDG,L,LL,
        EPE,LLL,DL,IPR,IRES,K,EP,
        I,II,LL1,DLK)
        BINARY FIXED,
        (OPT,CHECK,ERROR EXTERNAL)
        CHARACTER(1),.
   IF ETA NE 0                     /*PRESET ERROR INDICATOR       */
   THEN CHECK='A',.                /*A= ACCURACY NOT REACHED      */
   ELSE CHECK='0',.                /*0= SUCCESSFUL OPERATION      */
   IP1  =IP+1,.
   IF IP1 LE 1
   THEN DO,.
        CHECK='D',.                /*ERROR IN SPECIFIED DIMENSION */
        GO TO OUT,.
        END,.
   EP   =IP*IP1/2,.                /*SET UP ADDRESSING CONSTANTS  */
   EPE  =EP+IP1,.
   WE   =WORK(EPE),.
   IF CHECK='A'                    /*SET TEST TO ABSOLUTE VALUE OF*/
   THEN TEST =ABS(ETA*WE),.        /*SPEC. ACCURACY FOR WANTED FIT*/
   IPR,LL=0,.
   L,LL1=1,.
                                   /******************************/
        DO I =1 TO IP,.            /*FACTORIZE GIVEN MATRIX       */
        LL   =LL+I,.               /******************************/
        K    =0,.
ITER..                             /*COMPUTE ELEMENTS OF I-TH ROW */
        S    =0,.
             DO II=LL1 TO LL-1,.   /*MODIFY ELEMENTS IN I-TH      */
             S    =S+MULTIPLY(     /*ROW BY SCALAR PRODUCT OF     */
                  WORK(II),        /*ELEMENTS OF FACTORIZATION    */
                  WORK(L),53),.    /*IN ROW AND COLUMN CROSSING   */
             L    =L+1,.           /*AT CURRENT ELEMENT           */
             END,.
        R    =WORK(L),.
        S    =R-S,.
        IF L =LL                   /*TEST FOR LOSS OF SIGNIFICANCE*/
        THEN DO,.                  /*IN PIVOTAL DIVISOR           */
             IF S LE ABS(EPS*R)
             THEN DO,.
                  CHECK='P',.      /*MARK LOSS OF SIGNIFICANCE    */
                  GO TO SOL,.      /*BYPASS FURTHER FACTORIZATION */
                  END,.
             Q,S  =SQRT(S),.       /*CALCULATE DIAGONAL ELEMENT   */
             END,.                 /*OF FACTORIZATION             */
        ELSE S    =S/Q,.
        WORK(L)=S,.                /*STORE FINAL ELEMENT          */
        K    =K+1,.                /*OF FACTORIZATION             */
        L    =L+K,.
        IF K+I LE IP1              /*TEST IF ALL ELEMENTS OF I-TH */
        THEN GO TO ITER,.          /*ROW ARE COMPUTED             */
        LL1,L=LL+1,.
        WE   =WE-S*S,.
        AUX(I)=WE,.                /*STORE SQUARESUM OF RESIDUALS */
        IF CHECK='A'               /*TEST ON SPECIFIED PRECISION  */
        THEN IF WE LT TEST
        THEN DO,.
             CHECK='0',.           /*SUCCESSFUL OPERATION         */
             GO TO SOL,.           /*RESP. ETA ACCURACY REACHED   */
             END,.
        IPR  =IPR+1,.
        END,.                      /*END OF FACTORIZATION         */
   IF OPT='F'
   THEN GO TO OUT,.
   LL   =EPE,.                     /******************************/
                                   /*COMPUTE LEAST SQUARE FIT(S)  */
SOL..                              /******************************/
   RS   =EP+IPR,.                  /*INIT. ADDRESS RIGHT HAND SIDE*/
   DG   =LL-1-IPR,.                /*INIT. ADDRESS DIAGONAL TERM  */
        DO I =IPR TO 1 BY -1,.
        Q    =WORK(DG),.           /*SET Q TO I-TH DIAGONAL TERM  */
        R    =WORK(RS),.           /*SET R TO I-TH RIGHT HAND SIDE*/
        WORK(RS)=AUX(I),.          /*INSERT I-TH RESIDUAL         */
        RS   =RS-1,.
        DG   =DG-I,.
        LL,L =LL-1,.
        K    =IPR-I,.
        DL,DLK=IPR,.
REP..                              /*CALCULATE THE I-TH ELEMENT   */
        L,LLL=L-DL,.               /*FOR THE HIGHEST FIT AND      */
        DL,DLK=DL-1,.              /*OPTIONALLY OF ALL LOWER FITS */
        S    =0,.
             DO II=L+K TO L+1 BY -1,. /*FORM SCALAR PRODUCTS NEEDED  */
             S    =S+MULTIPLY(     /*WITH BACK SUBSTITUTION       */
                  WORK(LLL),
                  WORK(II),53),.
             LLL  =LLL-DLK,.
             DLK  =DLK-1,.
             END,.
        WORK(L)=(R-S)/Q,.
        K    =K-1,.
        IF OPT='A'                 /*REPEAT IF ALL FITS SHOULD    */
        THEN IF K GE 0             /*BE CALCULATED                */
        THEN GO TO REP,.
        END,.
OUT..
   IRES =IPR,.
   ERROR=CHECK,.
   END,.                           /*END OF PROCEDURE ASN         */
```

Purpose:

ASN computes the solution of normal equations set up by procedures APC1, APC2, and APLL.

Usage:

CALL ASN (WORK, IP, IRES, OPT, EPS, ETA);

WORK ((IP+1) (IP+2)/2) -

BINARY FLOAT [(53)]
Given vector, containing the lower triangular part of a symmetric coefficient matrix of normal equations, stored rowwise, followed by the right-hand side and the square sum of function values.
Resultant vector containing (sequentially) the coefficient vectors of computed least square fits, degree one up to IRES.
WORK((IP(IP+1)/2) + K), K=1, ..., IRES contains the residuals corresponding to the approximation fit of degree K.
If only the approximation fit of highest degree (that is, degree IRES) is calculated, the coefficient vector has the same storage allocation as if all fits were calculated (similarly for the corresponding residual vector).

IP - BINARY FIXED
Given number of fundamental functions.

IRES - BINARY FIXED
Resultant (highest) degree of approximation fit(s) with respect to the user-specified accuracy.

OPT - CHARACTER(1)
Given option for operations to be performed.

EPS - BINARY FLOAT [(53)]
Given relative tolerance for test on loss of significance.

ETA - BINARY FLOAT [(53)]
Given relative tolerance for tolerated square sum of residuals.

Remarks:

1. All operations are performed with respect to the user-specified tolerances EPS and ETA.

2. If OPT is not equal to 'A' or 'F', then ASN computes the least square fit of degree IRES only.
OPT='A' means all fits of degree one up to IRES are calculated.
OPT='F' means the given coefficient matrix A is factored in the form T*T, in the linear array WORK. The triangular matrix T is allocated in the same way as the upper (lower) triangular part of A. The right-hand side R is replaced by $(T^*)^{-1}R$.

3. For EPS a sensible value is between 10^{-3} and 10^{-6} (10^{-10} and 10^{-15}) in single (double) precision. The absolute tolerance used internally for the test on loss of significance is ABS (EPS times current pivotal element).

For ETA a realistic value is between 1 and 10^{-6} (1 and 10^{-15}) in single (double) precision. Nevertheless, ETA may be set equal to zero. If no specification is made for ETA, it is equivalent to setting ETA=0. The absolute tolerance used internally for the square sum of residuals is ABS (ETA times square sum of function values).

4. Let:

n_1 = maximal dimension for which no loss of significance was indicated during factorization

n_2 = smallest dimension for which the square sum of residuals does not exceed the absolute tolerance ETA

IRES is given by MIN (n_1, n_2, IP). (n_2 = IP for ETA = 0).

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

5. ERROR='D' means IP is less than 1.

ERROR='A' means respective ETA accuracy is not reached.

ERROR='P' means loss of significance was detected.

Method:

Calculation of the least square fits is done using Cholesky's square root method for symmetric factorization.

Mathematical Background:

Let $f(x), g_i(x)$, $i = 1, 2, \ldots, m$, and $w(x) > 0$ be functions defined for $x = x_1, x_2, \ldots, x_n$. The problem is to determine the coefficients c_i of the linear combination

$$p(x) = \sum_{i=1}^{m} c_i g_i(x) \text{ such that}$$

$$e_m = \sum_{k=1}^{n} w(x_k)\,(f(x_k) - p(x_k))^2 = \min. \qquad (1)$$

The necessary conditions

$$\frac{\partial e_m}{\partial c_i} = 0,\ i = 1, 2, \ldots, m \qquad (2)$$

form a system of m linear equations in m unknowns c_i.

To simplify the notation we introduce the following matrices:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad F = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix},$$

$$W = \begin{bmatrix} w(x_1) & & & \\ & w(x_2) & & \\ & & \ddots & \\ & & & w(x_n) \end{bmatrix},$$

$$c = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix}, \quad G = \begin{bmatrix} g_1(x_1) & \cdots & g_1(x_n) \\ \vdots & & \vdots \\ g_m(x_1) & \cdots & g_m(x_n) \end{bmatrix}$$

Then we have

$$e_m = (F^T - C^T G)\, W(F - G^T C)$$

or, with $e_0 = F^T WF$,

$$e_m = e_0 - 2C^T GWF + C^T GWG^T C \qquad (1')$$

Using equation (1'), the equations (2) may be written

$$GWG^T{}_C = GWF \qquad (2')$$

Combining (1') and (2') gives

$$e_0 - e_m = C^T GWG^T C \qquad (3)$$

The normal equations (2') for the unknown vector C may be solved using Cholesky's method since the coefficient matrix $A = GWG^T$ is obviously symmetric and it is positive definite if all the fundamental functions $g_i(x)$ are linearly independent for the arguments x_i -- that is, if the rows of G are linearly independent. Let R = GWF. Using Cholesky's method, A and R are replaced without additional storage requirements by T and $(T^T)^{-1}R$, where $A = T^T T$ and T is upper triangular.

An easy calculation shows

$$e_0 - e_m = \left\| (T^T)^{-1} R \right\|^2$$

Introducing additional fundamental functions in the linear combination p(x) will not affect the first m rows and columns of A or the first m elements of R. Therefore, Cholesky's method gives a decomposition of $e_0 - e_m$ into the separate components corresponding to individual degrees of freedom.

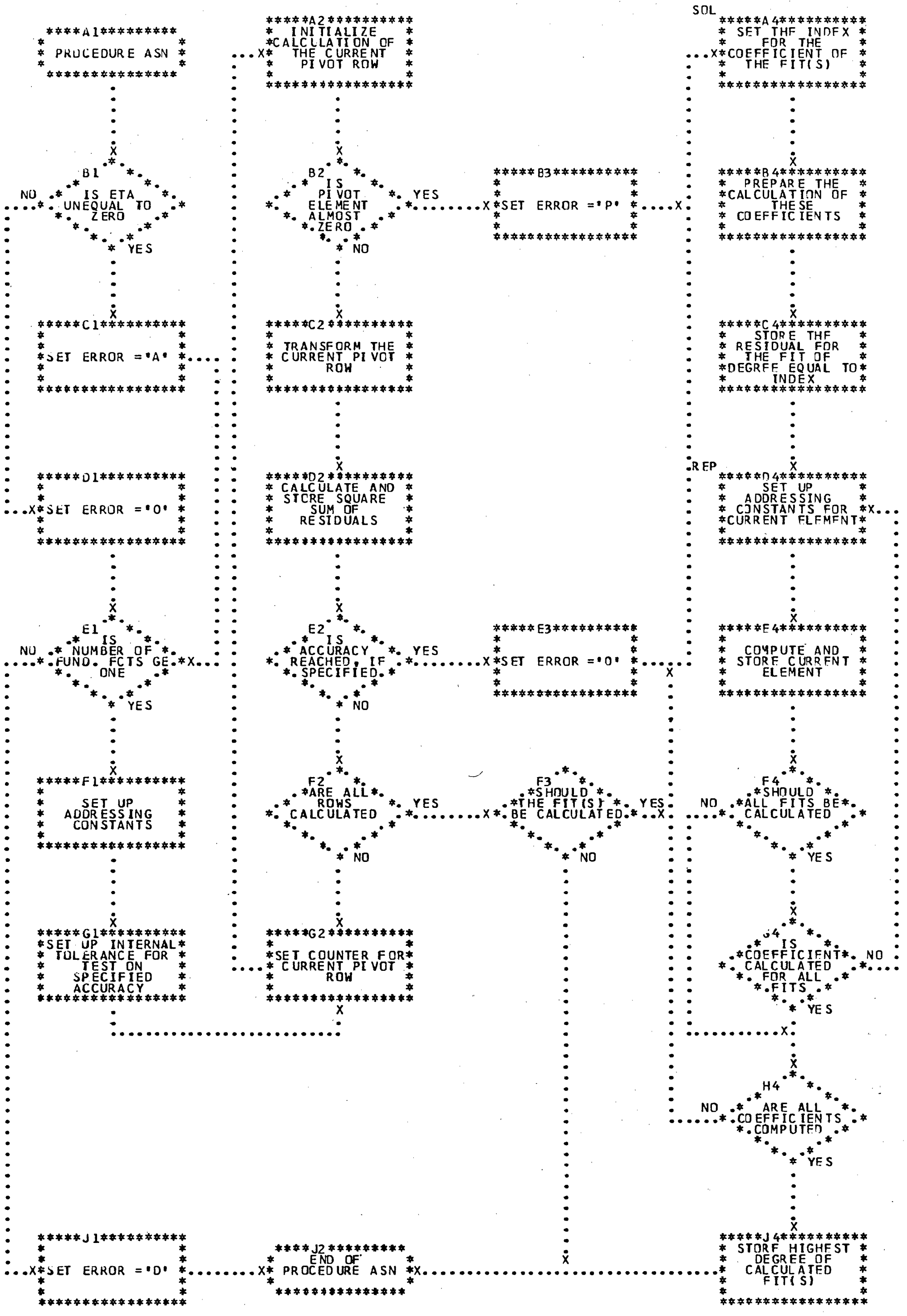
Programming Considerations:

All least squares fits of dimension 1, 2, ..., m may be computed from the reduced normal equations $TC = (T^T)^{-1}R$. If the solutions are generated in the storage locations of T, there is no additional storage requirement.

Using the decomposition of $e_0 - e_m$, the factorization may be terminated with dimension k if $e_k < \eta e_0$, giving the least squares fit of dimension k that satisfies the user-specified precision (relative tolerance η). Because of rounding errors this will work only if η is approximately between 1.0 and 1.0 E-6 in single precision, and between 1.0 and 1.0 E-15 in double precision. Nevertheless, the square sum of residuals corresponding to a least squares fit calculated in single (double) precision may be as small as $e_0\ 10^{-12}$ ($e_0\ 10^{-30}$).

Because of rounding errors the square root method may break down if very small or negative pivot elements indicate a loss of significance. Therefore, all pivot elements are tested against the absolute value of EPS times the current diagonal element of A. If the k-th pivot element is not greater than this internal test value, the normal equations are treated as if they had dimension k-1 only.

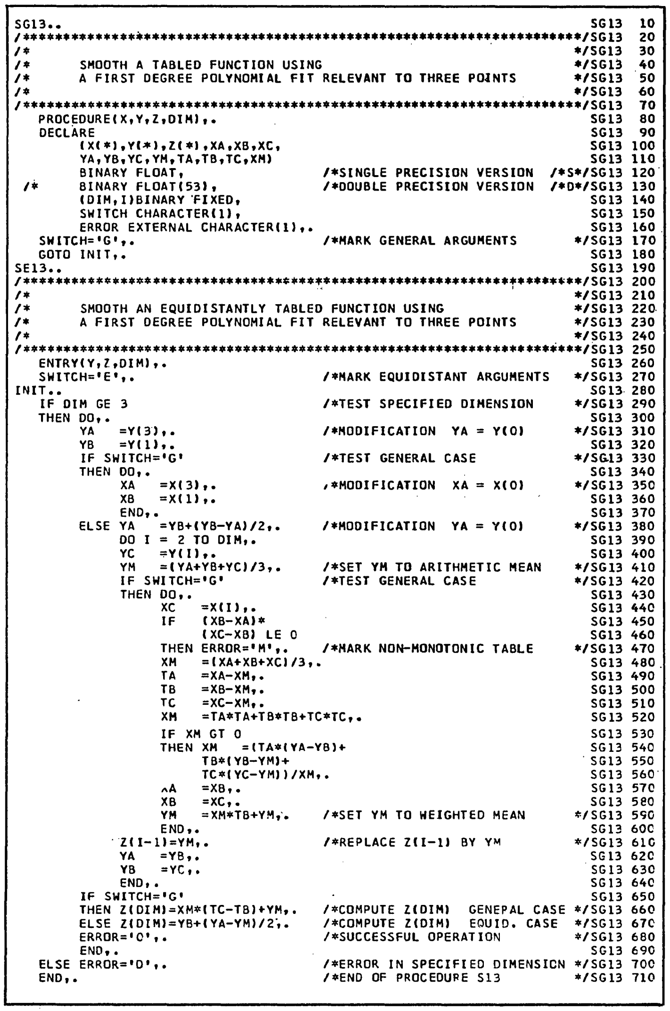
PROCEDURE ASN COMPUTES THE SOLUTION OF THE NCRMAL EQUATIONS SET UP BY PROCEDURES APC1, APC2 AND APLL



A1
PROCEDURE ASN
A2
INITIALIZE CALCULATION OF THE CURRENT PIVOT ROW
SOL
A4
SET THE INDEX FOR THE COEFFICIENT OF THE FIT(S)
B1
IS ETA UNEQUAL TO ZERO
NO
YES
B2
IS PIVOT ELEMENT ALMOST ZERO
YES
NO
B3
SET ERROR ='P'
B4
PREPARE THE CALCULATION OF THESE COEFFICIENTS
C1
SET ERROR ='A'
C2
TRANSFORM THE CURRENT PIVOT ROW
C4
STORE THE RESIDUAL FOR THE FIT OF DEGREE EQUAL TO INDEX
D1
SET ERROR ='O'
D2
CALCULATE AND STORE SQUARE SUM OF RESIDUALS
REP
D4
SET UP ADDRESSING CONSTANTS FOR CURRENT ELEMENT
E1
IS NUMBER OF FUND. FCTS GE ONE
NO
YES
E2
IS ACCURACY REACHED, IF SPECIFIED
YES
NO
E3
SET ERROR ='O'
E4
COMPUTE AND STORE CURRENT ELEMENT
F1
SET UP ADDRESSING CONSTANTS
F2
ARE ALL ROWS CALCULATED
YES
NO
F3
SHOULD THE FIT(S) BE CALCULATED
YES
NO
F4
SHOULD ALL FITS BE CALCULATED
NO
YES
G1
SET UP INTERNAL TOLERANCE FOR TEST ON SPECIFIED ACCURACY
G2
SET COUNTER FOR CURRENT PIVOT ROW
G4
IS COEFFICIENT CALCULATED FOR ALL FITS
NO
YES
H4
ARE ALL COEFFICIENTS COMPUTED
NO
YES
J1
SET ERROR ='D'
J2
END OF PROCEDURE ASN
J4
STORE HIGHEST DEGREE OF CALCULATED FIT(S)

Smoothing of Tabulated Functions

- Subroutine SG13/SE13



```
SG13..
/********************************************************************/
/*                                                                   */
/*      SMOOTH A TABLED FUNCTION USING                               */
/*      A FIRST DEGREE POLYNOMIAL FIT RELEVANT TO THREE POINTS       */
/*                                                                   */
/********************************************************************/
   PROCEDURE(X,Y,Z,DIM),.
   DECLARE
        (X(*),Y(*),Z(*),XA,XB,XC,
        YA,YB,YC,YM,TA,TB,TC,XM)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (DIM,I)BINARY FIXED,
        SWITCH CHARACTER(1),
        ERROR EXTERNAL CHARACTER(1),.
   SWITCH='G',.                       /*MARK GENERAL ARGUMENTS       */
   GOTO INIT,.
SE13..
/********************************************************************/
/*                                                                   */
/*      SMOOTH AN EQUIDISTANTLY TABLED FUNCTION USING                */
/*      A FIRST DEGREE POLYNOMIAL FIT RELEVANT TO THREE POINTS       */
/*                                                                   */
/********************************************************************/
   ENTRY(Y,Z,DIM),.
   SWITCH='E',.                       /*MARK EQUIDISTANT ARGUMENTS   */
INIT..
   IF DIM GE 3                        /*TEST SPECIFIED DIMENSION     */
   THEN DO,.
        YA   =Y(3),.                  /*MODIFICATION  YA = Y(0)      */
        YB   =Y(1),.
        IF SWITCH='G'                 /*TEST GENERAL CASE            */
        THEN DO,.
             XA   =X(3),.             ,*MODIFICATION  XA = X(0)      */
             XB   =X(1),.
             END,.
        ELSE YA   =YB+(YB-YA)/2,.     /*MODIFICATION  YA = Y(0)      */
             DO I = 2 TO DIM,.
             YC   =Y(I),.
             YM   =(YA+YB+YC)/3,.     /*SET YM TO ARITHMETIC MEAN    */
             IF SWITCH='G'            /*TEST GENERAL CASE            */
             THEN DO,.
                  XC   =X(I),.
                  IF   (XB-XA)*
                       (XC-XB) LE 0
                  THEN ERROR='M',.    /*MARK NON-MONOTONIC TABLE     */
                  XM   =(XA+XB+XC)/3,.
                  TA   =XA-XM,.
                  TB   =XB-XM,.
                  TC   =XC-XM,.
                  XM   =TA*TA+TB*TB+TC*TC,.
                  IF XM GT 0
                  THEN XM   =(TA*(YA-YB)+
                       TB*(YB-YM)+
                       TC*(YC-YM))/XM,.
                  ^A   =XB,.
                  XB   =XC,.
                  YM   =XM*TB+YM,.    /*SET YM TO WEIGHTED MEAN      */
                  END,.
             Z(I-1)=YM,.              /*REPLACE Z(I-1) BY YM         */
             YA   =YB,.
             YB   =YC,.
             END,.
        IF SWITCH='G'
        THEN Z(DIM)=XM*(TC-TB)+YM,.   /*COMPUTE Z(DIM)  GENEPAL CASE */
        ELSE Z(DIM)=YB+(YA-YM)/2,.    /*COMPUTE Z(DIM)  EQUID. CASE  */
        ERROR='0',.                   /*SUCCESSFUL OPERATION         */
        END,.
   ELSE ERROR='D',.                   /*ERROR IN SPECIFIED DIMENSION */
   END,.                              /*END OF PROCEDURE S13         */
```

Purpose:

SG13, SE13 computes a vector $Z = (z_1, \ldots, z_{DIM})$ of smoothed function values. SE13 requires a vector $Y = (y_1, \ldots, y_{DIM})$ and in the case of SG13 a vector $X = (x_1, \ldots, x_{DIM})$ of argument values must be given in addition. y_i corresponds to x_i, in the case of SE13 the y components correspond to equidistantly spaced argument values x_i, assuming $x_i - x_{i-1} = h$.

Usage:

CALL SG13 (X, Y, Z, DIM);

X(DIM) - BINARY FLOAT [(53)]
Given vector of argument values.

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of smoothed function values.

DIM - BINARY FIXED
Given dimension of vectors X, Y and Z.

CALL SE13 (Y, Z, DIM);

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of smoothed function values.

DIM - BINARY FIXED
Given dimension of vectors Y, Z.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR = 'D' means DIM is less than three.

ERROR = 'M' indicates a non-monotonic argument table, that is, for some i $(x_i-x_{i-1})(x_{i+1}-x_i)$, is less than or equal to zero. Vectors Z and Y may be identically allocated, which means that the given function values are replaced by the resultant smoothed function values.

Method:

The smoothed function values z_i are obtained by evaluating the least squares polynomial of degree one at x_i relevant to three successive points.

For references see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 258-311.

Mathematical Background:

For i = 3, . . . , n we must find m_i and b_i such that

$$w_i (x) = m_i x + b_i \tag{1}$$

gives the least-squares fit to the points (x_{i-2}, y_{i-2}), (x_{i-1}, y_{i-1}), and (x_i, y_i). The problem, then, is to minimize

$$F(m_i, b_i) = \sum_{k=0}^{2} \left[w_i(x_{i-k}) - y_{i-k} \right]^2$$

This minimum will occur when

$$\frac{\partial F}{\partial b_i} = 0 \text{ and } \frac{\partial F}{\partial m_i} = 0 \tag{2}$$

Now

$$\left.\begin{aligned} &\frac{\partial F}{\partial b_i} = 2 \sum_{k=0}^{2} \left[w_i\,(x_{i-k}) - y_{i-k} \right] \\ &\text{and} \\ &\frac{\partial F}{\partial m_i} = 2 \sum_{k=0}^{2} x_{i-k} \left[w_i(x_{i-k}) - y_{i-k} \right] \end{aligned}\right\} \tag{3}$$

Solving equations (2) and (3) yields:

$$m_i = \frac{\displaystyle\sum_{k=0}^{2} x_{i-k}\, y_{i-k} - 1/3 \left(\sum_{k=0}^{2} x_{i-k} \right) \left(\sum_{k=0}^{2} y_{i-k} \right)}{\displaystyle\sum_{k=0}^{2} x^2_{\ i-k} - 1/3 \left(\sum_{k=0}^{2} x_{i-k} \right)^2} \tag{4}$$

and

$$b_i = \frac{1}{3} \sum_{k=0}^{2} \left(y_{i-k} - m_i x_{i-k} \right) \tag{5}$$

Letting:

$$\begin{aligned} \bar{y}_i &= \frac{1}{3} \sum_{k=0}^{2} y_{i-k} \\ \bar{x}_i &= \frac{1}{3} \sum_{k=0}^{2} x_{i-k} \\ t_{i,k} &= x_{i-k} - \bar{x}_i \text{ and } v_{i,k} = y_{i-k} - \bar{y}_i \end{aligned} \tag{6}$$

we may rewrite (4) and (5) as:

$$m_i = \frac{\displaystyle\sum_{k=0}^{2} t_{i,k}\, v_{i,k}}{\displaystyle\sum_{k=0}^{2} t^2_{\ i,k}} \tag{7}$$

and

$$b_i = \bar{y}_i - m_i\, \bar{x}_i \tag{8}$$

Using (8) in (1) gives

$$w_i(x) = m_i(x - \bar{x}_i) + \bar{y}_i$$

where m_i is as in (7).

The desired smoothed values z_i are given by:

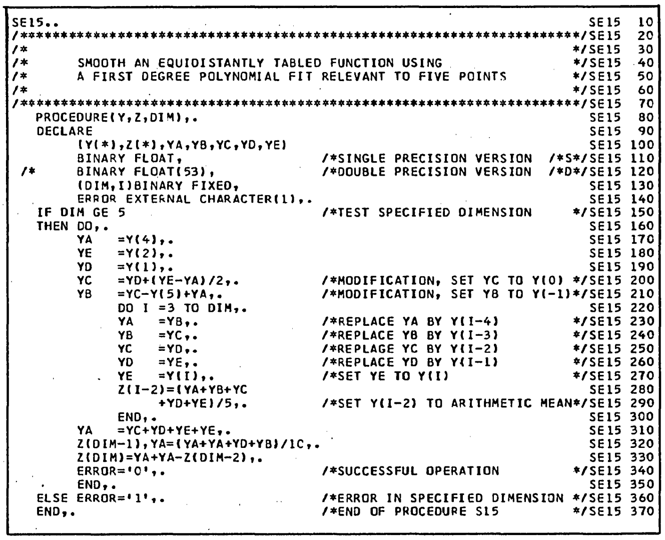
$$z_i = \begin{cases} w_3(x_1) = m_3 t_{3,2} + \bar{y}_3 & \text{if } i=1 \\ w_{i+1}(x_i) = m_{i+1} t_{i+1,1} + \bar{y}_{i+1} & \text{if } i=2,\ldots,n-1 \\ w_n(x_n) = m_n t_{n,0} + \bar{y}_n & \text{if } i=n \end{cases} \tag{9}$$

for generally tabulated argument values -- that is, for SG13.

In the case of equidistantly spaced argument values (that is, in case of SE13) we have the additional relation $x_i - x_{i-1} = h$, a constant, for $i = 2, \ldots, n$. This leads to the following expressions for the z_i:

$$z_i = \begin{cases} \frac{1}{6}(5y_1 + 2y_2 - y_3) & \text{if } i=1 \\ \frac{1}{3}(y_{i-1} + y_i + y_{i+1}) & \text{if } i=2,\ldots,n-1 \\ \frac{1}{6}(-y_{n-2} + 2y_{n-1} + 5y_n) & \text{if } i=n \end{cases} \tag{1}$$

- Subroutine SE15



```
SE15..                                                                 SE15  10
/*********************************************************************/SE15  20
/*                                                                   */SE15  30
/*      SMOOTH AN EQUIDISTANTLY TABLED FUNCTION USING                */SE15  40
/*      A FIRST DEGREE POLYNOMIAL FIT RELEVANT TO FIVE POINTS        */SE15  50
/*                                                                   */SE15  60
/*********************************************************************/SE15  70
   PROCEDURE(Y,Z,DIM),.                                                SE15  80
   DECLARE                                                             SE15  90
        (Y(*),Z(*),YA,YB,YC,YD,YE)                                     SE15 100
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/SE15 110
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/SE15 120
        (DIM,I)BINARY FIXED,                                           SE15 130
        ERROR EXTERNAL CHARACTER(1),.                                  SE15 140
   IF DIM GE 5                        /*TEST SPECIFIED DIMENSION     */SE15 150
   THEN DO,.                                                           SE15 160
        YA   =Y(4),.                                                   SE15 170
        YE   =Y(2),.                                                   SE15 180
        YD   =Y(1),.                                                   SE15 190
        YC   =YD+(YE-YA)/2,.          /*MODIFICATION, SET YC TO Y(0) */SE15 200
        YB   =YC-Y(5)+YA,.            /*MODIFICATION, SET YB TO Y(-1)*/SE15 210
             DO I =3 TO DIM,.                                          SE15 220
             YA   =YB,.               /*REPLACE YA BY Y(I-4)         */SE15 230
             YB   =YC,.               /*REPLACE YB BY Y(I-3)         */SE15 240
             YC   =YD,.               /*REPLAGE YC BY Y(I-2)         */SE15 250
             YD   =YE,.               /*REPLACE YD BY Y(I-1)         */SE15 260
             YE   =Y(I),.             /*SET YE TO Y(I)               */SE15 270
             Z(I-2)=(YA+YB+YC                                          SE15 280
                  +YD+YE)/5,.         /*SET Y(I-2) TO ARITHMETIC MEAN*/SE15 290
             END,.                                                     SE15 300
        YA   =YC+YD+YE+YE,.                                            SE15 310
        Z(DIM-1),YA=(YA+YA+YD+YB)/1C,.                                 SE15 320
        Z(DIM)=YA+YA-Z(DIM-2),.                                        SE15 330
        ERROR='0',.                   /*SUCCESSFUL OPERATION         */SE15 340
        END,.                                                          SE15 350
   ELSE ERROR='1',.                   /*ERROR IN SPECIFIED DIMENSION */SE15 360
   END,.                              /*END OF PROCEDURE S15         */SE15 370
```

Purpose:

SE15 computes a vector $Z = (z_1, z_2, \ldots, z_{DIM})$ of smoothed function values, given a vector $Y = (y_1, y_2, \ldots, y_{DIM})$ of function values whose components y_i correspond to DIM equidistantly spaced argument values x_i with $x_i - x_{i-1} = h$ for $i = 2, \ldots,$ DIM.

Usage:

CALL SE15 (Y, Z, DIM);

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of smoothed function values.

DIM - BINARY FIXED
Given dimension of vectors Y and Z.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR='1' means DIM is less than five.
Vectors Z and Y may be identically allocated, which means that the given function values are replaced by the resultant smoothed function values.

Method:

The smoothed function values are obtained by evaluation of the least squares polynomial of degree one relevant to five successive points.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 295-302.

Mathematical Background:

For $i = 5, \ldots, n$ we find m_i and b_i such that

$$w_i(x) = m_ix + b_i \tag{1}$$

gives the least-squares fit to the points (x_{i-k}, y_{i-k}), $k=0, \ldots, 4$. The problem, then, is to minimize

$$F(m_i, b_i) = \sum_{k=0}^{4} \left[w_i(x_{i-k}) - y_{i-k} \right]^2$$

This minimum will occur when

$$\frac{\partial F}{\partial b_i} = 0 \text{ and } \frac{\partial F}{\partial m_i} = 0 \tag{2}$$

Now

$$\frac{\partial F}{\partial b_i} = 2\sum_{k=0}^{4} \left[w_i(x_{i-k}) - y_{i-k} \right]$$

and

$$\frac{\partial F}{\partial m_i} = 2\sum_{k=0}^{4} x_{i-k} \left[w_i(x_{i-k}) - y_{i-k} \right] \tag{3}$$

Solving equations (2) and (3) yields:

$$m_i = \frac{\sum_{k=0}^{4} x_{i-k}y_{i-k} - \frac{1}{5}\left(\sum_{k=0}^{4} x_{i-k}\right)\left(\sum_{k=0}^{4} y_{i-k}\right)}{\sum_{k=0}^{4} x^2_{i-k} - \frac{1}{5}\left(\sum_{k=0}^{4} x_{i-k}\right)^2} \tag{4}$$

and

$$b_i = \frac{1}{5} \sum_{k=0}^{4} \left[y_{i-k} - m_ix_i-k \right] \tag{5}$$

Using the fact that $x_j - x_{j-1} = h$, a constant, for $j = 2,\dots,n$, equations (4) and (5) may be rewritten as

$$m_i = \frac{1}{10h}\,(2y_i + y_{i-1} - y_{i-3} - 2y_{i-4}) \qquad (6)$$

and

$$b_i = \frac{1}{5}\sum_{k=0}^{4} y_{i-k} - m_i x_{i-2} \qquad (7)$$

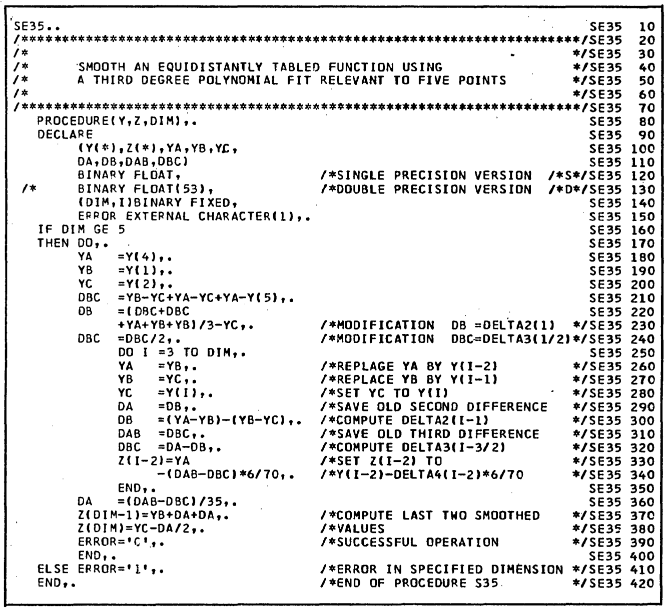
Using equation (7) in equation (1) yields

$$w_i(x) = m_i(x - x_{i-2}) + \frac{1}{5}\,(y_{i-4} + \dots + y_i)$$

The desired smoothed function values z_i are given by:

$$z_i = \begin{cases} w_5(x_1) = \frac{1}{5}\,(3y_1 + 2y_2 + y_3 - y_5) & i=1 \\ w_5(x_2) = \frac{1}{10}\,(4y_1 + 3y_2 + 2y_3 + y_4) & i=2 \\ w_{i+2}(x_i) = \frac{1}{5}\,(y_{i-2} + y_{i-1} + y_i + y_{i+1} + y_{i+2}) & i=3,\dots,n-2 \\ w_n(x_{n-1}) = \frac{1}{10}\,(y_{n-3} + 2y_{n-2} + 3y_{n-1} + 4y_n) & i=n-1 \\ w_n(x_n) = \frac{1}{5}\,(-y_{n-4} + y_{n-2} + 2y_{n-1} + 3y_n) & i=n \end{cases} \qquad (8)$$

- <u>Subroutine</u> SE35



```
SE35..                                                                 SE35  10
/**********************************************************************/SE35  20
/*                                                                    */SE35  30
/*     SMOOTH AN EQUIDISTANTLY TABLED FUNCTION USING                  */SE35  40
/*     A THIRD DEGREE POLYNOMIAL FIT RELEVANT TO FIVE POINTS          */SE35  50
/*                                                                    */SE35  60
/**********************************************************************/SE35  70
   PROCEDURE(Y,Z,DIM),.                                                SE35  80
   DECLARE                                                             SE35  90
        (Y(*),Z(*),YA,YB,YC,                                           SE35 100
        DA,DB,DAB,DBC)                                                 SE35 110
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/SE35 120
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/SE35 130
        (DIM,I)BINARY FIXED,                                           SE35 140
        ERROR EXTERNAL CHARACTER(1),.                                  SE35 150
   IF DIM GE 5                                                         SE35 160
   THEN DO,.                                                           SE35 170
        YA   =Y(4),.                                                   SE35 180
        YB   =Y(1),.                                                   SE35 190
        YC   =Y(2),.                                                   SE35 200
        DBC  =YB-YC+YA-YC+YA-Y(5),.                                    SE35 210
        DB   =(DBC+DBC                                                 SE35 220
             +YA+YB+YB)/3-YC,.         /*MODIFICATION  DB =DELTA2(1)  */SE35 230
        DBC  =DBC/2,.                  /*MODIFICATION  DBC=DELTA3(1/2)*/SE35 240
             DO I =3 TO DIM,.                                          SE35 250
             YA   =YB,.                /*REPLAGE YA BY Y(I-2)         */SE35 260
             YB   =YC,.                /*REPLACE YB BY Y(I-1)         */SE35 270
             YC   =Y(I),.              /*SET YC TO Y(I)               */SE35 280
             DA   =DB,.                /*SAVE OLD SECOND DIFFERENCE   */SE35 290
             DB   =(YA-YB)-(YB-YC),.   /*COMPUTE DELTA2(I-1)          */SE35 300
             DAB  =DBC,.               /*SAVE OLD THIRD DIFFERENCE    */SE35 310
             DBC  =DA-DB,.             /*COMPUTE DELTA3(I-3/2)        */SE35 320
             Z(I-2)=YA                 /*SET Z(I-2) TO                */SE35 330
                  -(DAB-DBC)*6/70,.    /*Y(I-2)-DELTA4(I-2)*6/70      */SE35 340
             END,.                                                     SE35 350
        DA   =(DAB-DBC)/35,.                                           SE35 360
        Z(DIM-1)=YB+DA+DA,.            /*COMPUTE LAST TWO SMOOTHED    */SE35 370
        Z(DIM)=YC-DA/2,.               /*VALUES                       */SE35 380
        ERROR='C',.                    /*SUCCESSFUL OPERATION         */SE35 390
        END,.                                                          SE35 400
   ELSE ERROR='1',.                    /*ERROR IN SPECIFIED DIMENSION */SE35 410
   END,.                               /*END OF PROCEDURE S35         */SE35 420
```

Purpose:

SE35 computes a vector $Z = (z_1, z_2, \dots, z_{DIM})$ of smoothed function values, given a vector $Y = (y_1, y_2, \dots, y_{DIM})$ of function values whose components y_i correspond to DIM equidistantly spaced argument values x_i with $x_i - x_{i-1} = h$ for $i = 2, \dots,$ DIM.

Usage:

CALL SE35 (Y, Z, DIM);

Y(DIM) - BINARY FLOAT [(53)]
Given vector of function values.

Z(DIM) - BINARY FLOAT [(53)]
Resultant vector of smoothed function values.

DIM - BINARY FIXED
Given dimension of vector Y and Z.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected.

ERROR='1' means DIM is less than five.
Vectors Z and Y may be identically allocated, which means that the given function values are replaced by the resultant smoothed function values.

Method:

The smoothed function values z_i are obtained by evaluating at x_i the least squares polynomial of degree three relevant to five successive points.

For reference see:

F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York-Toronto-London, 1956, pp. 295-302.

Mathematical Background:

For $i = 5, \ldots, n$ we must find a_i, b_i, c_i, and d_i such that

$$w_i\,(x) = a_i x^3 + b_i x^2 + c_i x + d_i \qquad (1)$$

gives the least-squares fit to the points (x_{i-k}, y_{i-k}), $k = 0, \ldots, 4$.

The problem, thus, is to minimize

$$F(a_i,\, b_i,\, c_i,\, d_i) = \sum_{k=0}^{4} \left[w_i(x_{i-k}) - y_{i-k} \right]^2 \qquad (2)$$

The minimum will occur when

$$\frac{\partial F}{\partial a_i} = \frac{\partial F}{\partial b_i} = \frac{\partial F}{\partial c_i} = \frac{\partial F}{\partial d_i} = 0$$

Now:

$$\left.\begin{aligned}
\frac{\partial F}{\partial a_i} &= 2 \sum_{k=0}^{4} x_{i-k}^3 \left[w_i(x_{i-k}) - y_{i-k} \right] \\
\frac{\partial F}{\partial b_i} &= 2 \sum_{k=0}^{4} x_{i-k}^2 \left[w_i(x_{i-k}) - y_{i-k} \right] \\
\frac{\partial F}{\partial c_i} &= 2 \sum_{k=0}^{4} x_{i-k} \left[w_i(x_{i-k}) - y_{i-k} \right] \\
\frac{\partial F}{\partial d_i} &= 2 \sum_{k=0}^{4} \left[w_i(x_{i-k}) - y_{i-k} \right]
\end{aligned}\right\} \qquad (3)$$

Solving (2) and (3) for a_i, b_i, c_i, and d_i, with $x_i - x_{i-1} = h$, we get:

$$a_i = A_i$$

$$b_i = -3\, A_i\, x_{i-2} + B_i$$

$$c_i = 3 A_i x_{i-2}^2 - 2 B_i x_{i-2} + C_i$$

$$d_i = -A_i x_{i-2}^3 + B_i x_{i-2}^2 - C_i x_{i-2} + D_i + \bar{y}_i$$

where:

$$\bar{y}_i = \frac{1}{5} \sum_{k=0}^{4} y_{i-k}$$

$$A_i = -\frac{1}{12h^3}\,(y_{i-4} - 2y_{i-3} + 2y_{i-1} - y_i)$$

$$B_i = \frac{1}{14h^2}\,(4y_{i-4} + y_{i-3} + y_{i-1} + 4y_i - 10\bar{y}_i)$$

$$C_i = \frac{1}{12h}\,(y_{i-4} - 8y_{i-3} + 8y_{i-1} - y_i)$$

$$D_i = -\frac{1}{7}\,(4y_{i-4} + y_{i-3} + y_{i-1} + 4y_i - 10\bar{y}_i)$$

Finally, the desired smoothed values z_i are given by:

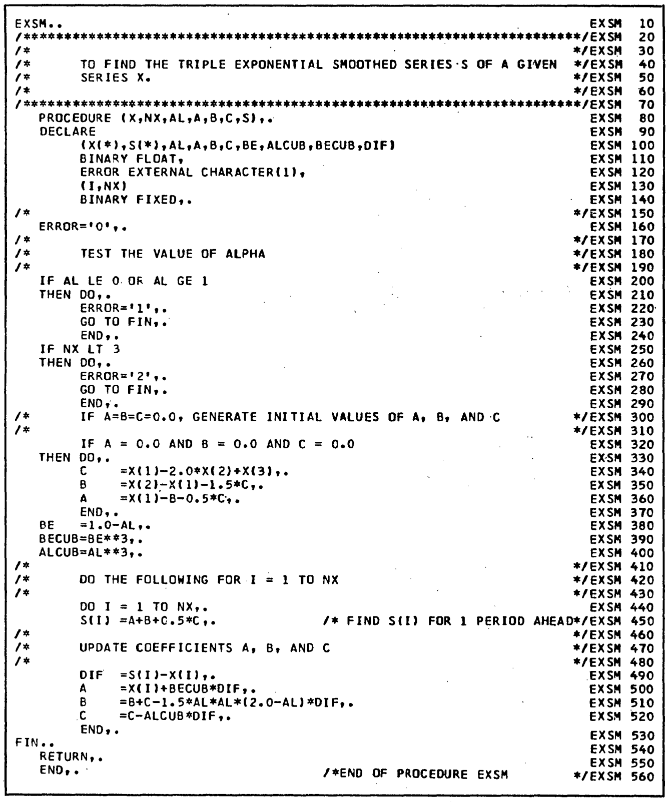
$$z_i = \left\{\begin{array}{lll}
w_5\,(x_1) & = y_1 - \frac{1}{70}\,\delta^4 y_3 & \text{if } i{=}1 \\
w_5\,(x_2) & = y_2 + \frac{2}{35}\,\delta^4 y_3 & \text{if } i{=}2 \\
w_{i+2}(x_i) & = y_i - \frac{3}{35}\,\delta^4 y_i & \text{if } i{=}3, \ldots, n{-}2 \\
w_n(x_{n-1}) & = y_{n-1} + \frac{2}{35}\,\delta^4 y_{n-2} & \text{if } i{=}n{-}1 \\
w_n(x_n) & = y_n - \frac{1}{70}\,\delta^4 y_{n-2} & \text{if } i{=}n
\end{array}\right\} \qquad (4)$$

where:

$$\delta^4 y_i = y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2}$$

for $i{=}3, \ldots, n{-}2$

- Subroutine EXSM



```
EXSM..
/**********************************************************************/
/*                                                                    */
/*      TO FIND THE TRIPLE EXPONENTIAL SMOOTHED SERIES S OF A GIVEN   */
/*      SERIES X.                                                     */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (X,NX,AL,A,B,C,S),.
   DECLARE
        (X(*),S(*),AL,A,B,C,BE,ALCUB,BECUB,DIF)
        BINARY FLOAT,
        ERROR EXTERNAL CHARACTER(1),
        (I,NX)
        BINARY FIXED,.
/*                                                                    */
   ERROR='0',.
/*                                                                    */
/*      TEST THE VALUE OF ALPHA                                       */
/*                                                                    */
   IF AL LE 0 OR AL GE 1
   THEN DO,.
        ERROR='1',.
        GO TO FIN,.
        END,.
   IF NX LT 3
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
/*      IF A=B=C=0.0, GENERATE INITIAL VALUES OF A, B, AND C          */
/*                                                                    */
        IF A = 0.0 AND B = 0.0 AND C = 0.0
   THEN DO,.
        C    =X(1)-2.0*X(2)+X(3),.
        B    =X(2)-X(1)-1.5*C,.
        A    =X(1)-B-0.5*C,.
        END,.
   BE   =1.0-AL,.
   BECUB=BE**3,.
   ALCUB=AL**3,.
/*                                                                    */
/*      DO THE FOLLOWING FOR I = 1 TO NX                              */
/*                                                                    */
        DO I = 1 TO NX,.
        S(I) =A+B+0.5*C,.                /* FIND S(I) FOR 1 PERIOD AHEAD*/
/*                                                                    */
/*      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,.
        C    =C-ALCUB*DIF,.
        END,.
FIN..
   RETURN,.
   END,.                             /*END OF PROCEDURE EXSM          */
```

Purpose:

EXSM develops the triple exponential smoothed series S of the given series X.

Usage:

CALL EXSM (X, NX, AL, A, B, C, S);

Description of parameters:

X(NX) - BINARY FLOAT
Given vector containing time series data to be exponentially smoothed.

NX - BINARY FIXED
Given number of elements in X.

AL - BINARY FLOAT
Given smoothing constant alpha. AL must be greater than zero and less than one.

A, B, C- BINARY FLOAT
Given coefficients of the prediction equation where S is predicted T periods hence by

$$A + B \cdot T + C \cdot T^2/_2$$

As input: If A=B=C=0, the program will provide initial values. If at least one of A, B, and C is not zero, the program will take given values as initial values.

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

S(NX) - BINARY FLOAT
Resultant vector containing triple exponential smoothed time series.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error conditions that may be detected:

ERROR=1 - The specified smoothing constant, AL, is less than or equal to zero or is greater than or equal to one.

ERROR=2 - The number of data points is less than three.

Method:

Refer to R. G. Brown, <u>Smoothing, Forecasting and Prediction of Discrete Time Series</u>, Prentice-Hall, 1963, pp. 140 to 144.

Mathematical Background:

This procedure calculates a smoothed series $S_1, S_2, \ldots, S_{NX}$, given time series $X_1, X_2, \ldots, X_{NX}$ and a smoothing constant α. Also, at the end of the computation, the coefficients A, B, and C are given for the expression $A + B(T) + C(T)^2/_2$. This expression can be used to find estimates of the smoothed series a given number of time periods, T, ahead.

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

1. Finds S_i for one period ahead

$$S_i = A + B + 5C \tag{1}$$

2. Update coefficients A, B, and C

$$A = X_i + (1 - \alpha)^3 (S_i - X_i) \tag{2}$$

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

$$C = C - (\alpha^3) (S_i - X_i) \tag{4}$$

where α = smoothing constant specified by the user

$$(0.0 < \alpha < 1.0)$$

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

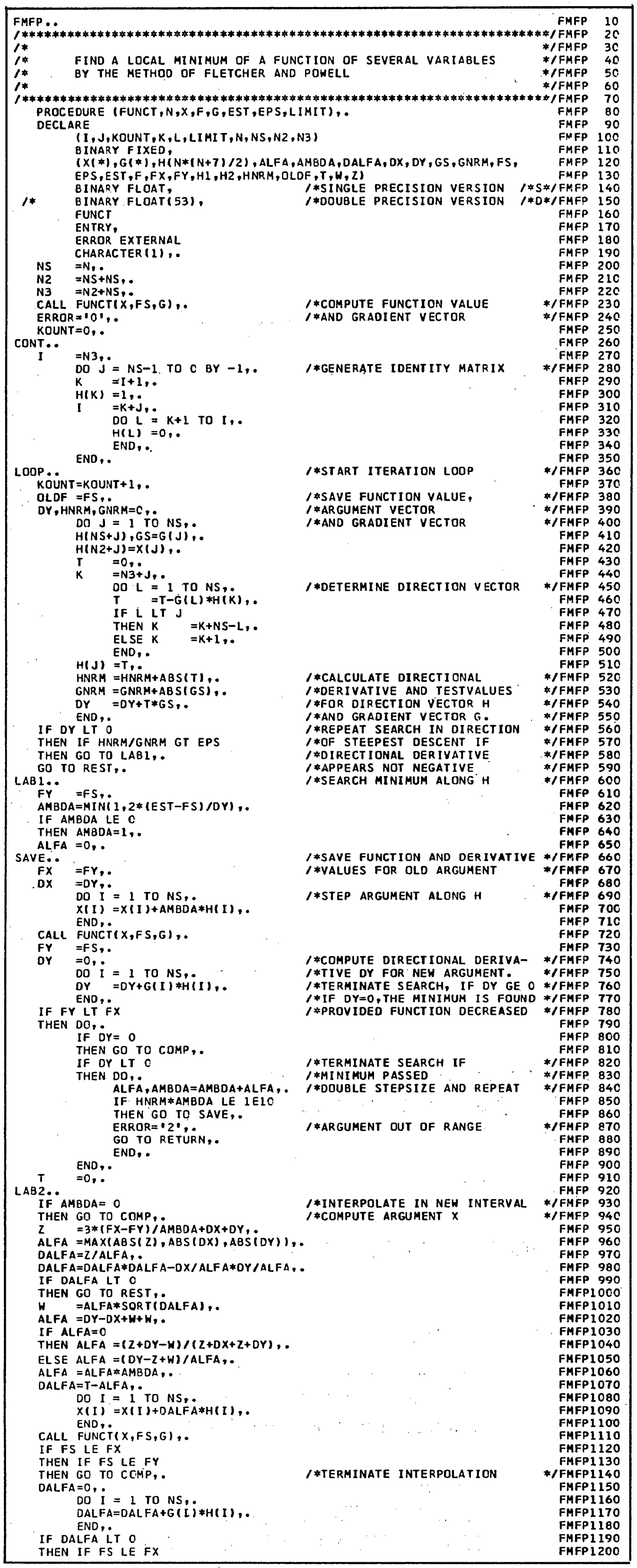
$$C = X_1 - 2X_2 + X_3 \qquad (5)$$

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

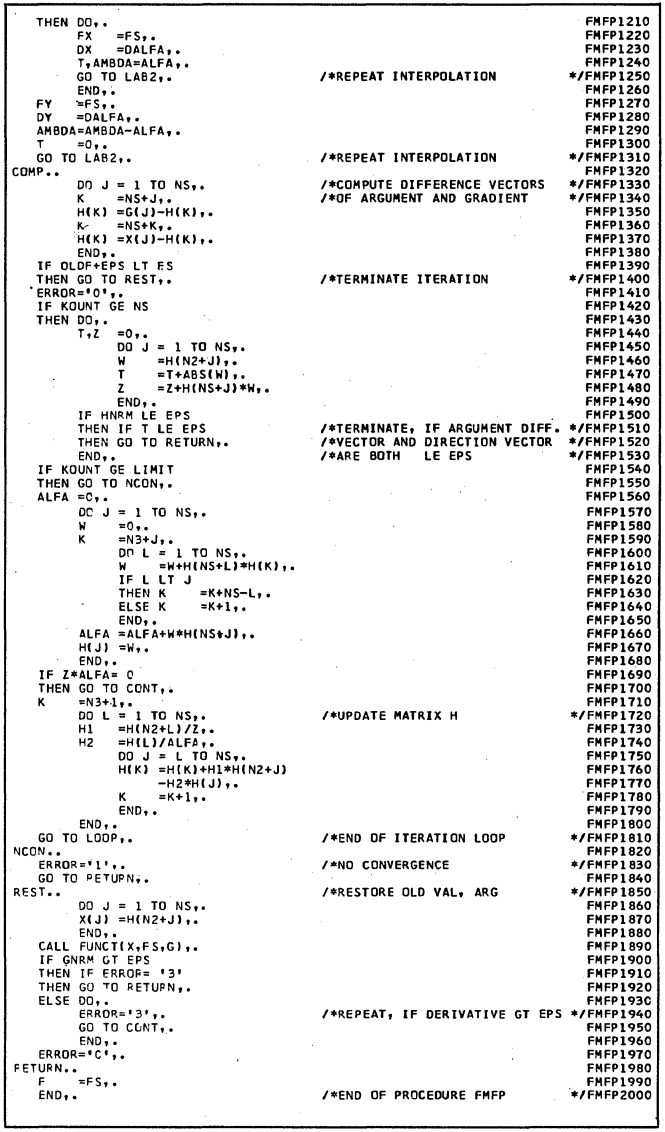
$$A = X_1 - B - 0.5C \qquad (7)$$

Roots and Extrema of Functions

- Subroutine FMFP



```
FMFP..
/*********************************************************************/
/*                                                                   */
/*      FIND A LOCAL MINIMUM OF A FUNCTION OF SEVERAL VARIABLES      */
/*      BY THE METHOD OF FLETCHER AND POWELL                         */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (FUNCT,N,X,F,G,EST,EPS,LIMIT),.
   DECLARE
        (I,J,KOUNT,K,L,LIMIT,N,NS,N2,N3)
        BINARY FIXED,
        (X(*),G(*),H(N*(N+7)/2),ALFA,AMBDA,DALFA,DX,DY,GS,GNRM,FS,
        EPS,EST,F,FX,FY,H1,H2,HNRM,OLDF,T,W,Z)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        FUNCT
        ENTRY,
        ERROR EXTERNAL
        CHARACTER(1),.
   NS   =N,.
   N2   =NS+NS,.
   N3   =N2+NS,.
   CALL FUNCT(X,FS,G),.               /*COMPUTE FUNCTION VALUE       */
   ERROR='0',.                        /*AND GRADIENT VECTOR          */
   KOUNT=0,.
CONT..
   I    =N3,.
        DO J = NS-1 TO 0 BY -1,.      /*GENERATE IDENTITY MATRIX     */
        K    =I+1,.
        H(K) =1,.
        I    =K+J,.
             DO L = K+1 TO I,.
             H(L) =0,.
             END,.
        END,.
LOOP..                                /*START ITERATION LOOP         */
   KOUNT=KOUNT+1,.
   OLDF =FS,.                         /*SAVE FUNCTION VALUE,         */
   DY,HNRM,GNRM=0,.                   /*ARGUMENT VECTOR              */
        DO J = 1 TO NS,.              /*AND GRADIENT VECTOR          */
        H(NS+J),GS=G(J),.
        H(N2+J)=X(J),.
        T    =0,.
        K    =N3+J,.
             DO L = 1 TO NS,.         /*DETERMINE DIRECTION VECTOR   */
             T    =T-G(L)*H(K),.
             IF L LT J
             THEN K    =K+NS-L,.
             ELSE K    =K+1,.
             END,.
        H(J) =T,.
        HNRM =HNRM+ABS(T),.           /*CALCULATE DIRECTIONAL        */
        GNRM =GNRM+ABS(GS),.          /*DERIVATIVE AND TESTVALUES    */
        DY   =DY+T*GS,.               /*FOR DIRECTION VECTOR H       */
        END,.                         /*AND GRADIENT VECTOR G.       */
   IF DY LT 0                         /*REPEAT SEARCH IN DIRECTION   */
   THEN IF HNRM/GNRM GT EPS           /*OF STEEPEST DESCENT IF       */
   THEN GO TO LAB1,.                  /*DIRECTIONAL DERIVATIVE       */
   GO TO REST,.                       /*APPEARS NOT NEGATIVE         */
LAB1..                                /*SEARCH MINIMUM ALONG H       */
   FY   =FS,.
   AMBDA=MIN(1,2*(EST-FS)/DY),.
   IF AMBDA LE 0
   THEN AMBDA=1,.
   ALFA =0,.
SAVE..                                /*SAVE FUNCTION AND DERIVATIVE */
   FX   =FY,.                         /*VALUES FOR OLD ARGUMENT      */
   DX   =DY,.
        DO I = 1 TO NS,.              /*STEP ARGUMENT ALONG H        */
        X(I) =X(I)+AMBDA*H(I),.
        END,.
   CALL FUNCT(X,FS,G),.
   FY   =FS,.
   DY   =0,.                          /*COMPUTE DIRECTIONAL DERIVA-  */
        DO I = 1 TO NS,.              /*TIVE DY FOR NEW ARGUMENT.    */
        DY   =DY+G(I)*H(I),.          /*TERMINATE SEARCH, IF DY GE 0 */
        END,.                         /*IF DY=0,THE MINIMUM IS FOUND */
   IF FY LT FX                        /*PROVIDED FUNCTION DECREASED  */
   THEN DO,.
        IF DY= 0
        THEN GO TO COMP,.
        IF DY LT 0                    /*TERMINATE SEARCH IF          */
        THEN DO,.                     /*MINIMUM PASSED               */
             ALFA,AMBDA=AMBDA+ALFA,.  /*DOUBLE STEPSIZE AND REPEAT   */
             IF HNRM*AMBDA LE 1E10
             THEN GO TO SAVE,.
             ERROR='2',.              /*ARGUMENT OUT OF RANGE        */
             GO TO RETURN,.
             END,.
        END,.
   T    =0,.
LAB2..
   IF AMBDA= 0                        /*INTERPOLATE IN NEW INTERVAL  */
   THEN GO TO COMP,.                  /*COMPUTE ARGUMENT X           */
   Z    =3*(FX-FY)/AMBDA+DX+DY,.
   ALFA =MAX(ABS(Z),ABS(DX),ABS(DY)),.
   DALFA=Z/ALFA,.
   DALFA=DALFA*DALFA-DX/ALFA*DY/ALFA,.
   IF DALFA LT 0
   THEN GO TO REST,.
   W    =ALFA*SQRT(DALFA),.
   ALFA =DY-DX+W+W,.
   IF ALFA=0
   THEN ALFA =(Z+DY-W)/(Z+DX+Z+DY),.
   ELSE ALFA =(DY-Z+W)/ALFA,.
   ALFA =ALFA*AMBDA,.
   DALFA=T-ALFA,.
        DO I = 1 TO NS,.
        X(I) =X(I)+DALFA*H(I),.
        END,.
   CALL FUNCT(X,FS,G),.
   IF FS LE FX
   THEN IF FS LE FY
   THEN GO TO COMP,.                  /*TERMINATE INTERPOLATION      */
   DALFA=0,.
        DO I = 1 TO NS,.
        DALFA=DALFA+G(I)*H(I),.
        END,.
   IF DALFA LT 0
   THEN IF FS LE FX
```



```
   THEN DO,.                                                          FMFP1210
        FX   =FS,.                                                    FMFP1220
        DX   =DALFA,.                                                 FMFP1230
        T,AMBDA=ALFA,.                                                FMFP1240
        GO TO LAB2,.                  /*REPEAT INTERPOLATION        */FMFP1250
        END,.                                                         FMFP1260
   FY   =FS,.                                                         FMFP1270
   DY   =DALFA,.                                                      FMFP1280
   AMBDA=AMBDA-ALFA,.                                                 FMFP1290
   T    =0,.                                                          FMFP1300
   GO TO LAB2,.                       /*REPEAT INTERPOLATION        */FMFP1310
COMP..                                                                FMFP1320
        DO J = 1 TO NS,.              /*COMPUTE DIFFERENCE VECTORS  */FMFP1330
        K    =NS+J,.                  /*OF ARGUMENT AND GRADIENT    */FMFP1340
        H(K) =G(J)-H(K),.                                             FMFP1350
        K    =NS+K,.                                                  FMFP1360
        H(K) =X(J)-H(K),.                                             FMFP1370
        END,.                                                         FMFP1380
   IF OLDF+EPS LT FS                                                  FMFP1390
   THEN GO TO REST,.                  /*TERMINATE ITERATION         */FMFP1400
   ERROR='0',.                                                        FMFP1410
   IF KOUNT GE NS                                                     FMFP1420
   THEN DO,.                                                          FMFP1430
        T,Z  =0,.                                                     FMFP1440
             DO J = 1 TO NS,.                                         FMFP1450
             W    =H(N2+J),.                                          FMFP1460
             T    =T+ABS(W),.                                         FMFP1470
             Z    =Z+H(NS+J)*W,.                                      FMFP1480
             END,.                                                    FMFP1490
        IF HNRM LE EPS                                                FMFP1500
        THEN IF T LE EPS              /*TERMINATE, IF ARGUMENT DIFF. */FMFP1510
        THEN GO TO RETURN,.           /*VECTOR AND DIRECTION VECTOR */FMFP1520
        END,.                         /*ARE BOTH   LE EPS           */FMFP1530
   IF KOUNT GE LIMIT                                                  FMFP1540
   THEN GO TO NCON,.                                                  FMFP1550
   ALFA =C,.                                                          FMFP1560
        DC J = 1 TO NS,.                                              FMFP1570
        W    =0,.                                                     FMFP1580
        K    =N3+J,.                                                  FMFP1590
             DO L = 1 TO NS,.                                         FMFP1600
             W    =W+H(NS+L)*H(K),.                                   FMFP1610
             IF L LT J                                                FMFP1620
             THEN K    =K+NS-L,.                                      FMFP1630
             ELSE K    =K+1,.                                         FMFP1640
             END,.                                                    FMFP1650
        ALFA =ALFA+W*H(NS+J),.                                        FMFP1660
        H(J) =W,.                                                     FMFP1670
        END,.                                                         FMFP1680
   IF Z*ALFA= C                                                       FMFP1690
   THEN GO TO CONT,.                                                  FMFP1700
   K    =N3+1,.                                                       FMFP1710
        DO L = 1 TO NS,.              /*UPDATE MATRIX H             */FMFP1720
        H1   =H(N2+L)/Z,.                                             FMFP1730
        H2   =H(L)/ALFA,.                                             FMFP1740
             DO J = L TO NS,.                                         FMFP1750
             H(K) =H(K)+H1*H(N2+J)                                    FMFP1760
                  -H2*H(J),.                                          FMFP1770
             K    =K+1,.                                              FMFP1780
             END,.                                                    FMFP1790
        END,.                                                         FMFP1800
   GO TO LOOP,.                       /*END OF ITERATION LOOP       */FMFP1810
NCON..                                                                FMFP1820
   ERROR='1',.                        /*NO CONVERGENCE              */FMFP1830
   GO TO PETUPN,.                                                     FMFP1840
REST..                                /*RESTORE OLD VAL, ARG        */FMFP1850
        DO J = 1 TO NS,.                                              FMFP1860
        X(J) =H(N2+J),.                                               FMFP1870
        END,.                                                         FMFP1880
   CALL FUNCT(X,FS,G),.                                               FMFP1890
   IF GNRM GT EPS                                                     FMFP1900
   THEN IF ERROF= '3'                                                 FMFP1910
   THEN GO TO RETUPN,.                                                FMFP1920
   ELSE DO,.                                                          FMFP1930
        ERROR='3',.                   /*REPEAT, IF DERIVATIVE GT EPS */FMFP1940
        GO TO CONT,.                                                  FMFP1950
        END,.                                                         FMFP1960
   ERROR='C',.                                                        FMFP1970
FETURN..                                                              FMFP1980
   F    =FS,.                                                         FMFP1990
   END,.                              /*END OF PROCEDURE FMFP       */FMFP2000
```

Purpose:

FMFP determines an unconstrained minimum of a function of several variables, given a starting value of argument vector.

Usage:

CALL FMFP (FUNCT, N, X, F, G, EST, EPS, LIMIT);

FUNCT - ENTRY
Given procedure computing function values and gradient vectors. This procedure must be supplied by the user.

Usage:
CALL FUNCT (X, FS, G);
X(N) - BINARY FLOAT [(53)]
Given n-dimensional argument vector.
FS - BINARY FLOAT [(53)]
Resultant function value.
G(N) - BINARY FLOAT [(53)]
Resultant gradient vector.

N - BINARY FIXED
Given number of variables (= dimension of argument vector).

X(N) - BINARY FLOAT [(53)]
Given starting value of argument vector. Resultant argument vector corresponding to the minimum.

F - BINARY FLOAT [(53)]
Resultant minimum function value.

G(N) - BINARY FLOAT [(53)]
Resultant gradient vector corresponding to the minimum.

EST - BINARY FLOAT [(53)]
Given estimate of minimum function value.

EPS - BINARY FLOAT [(53)]
Given test value representing the expected absolute error.

LIMIT - BINARY FIXED
Given maximum number of iterations to be performed.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR='1' means no convergence in LIMIT iterations.
ERROR='2' means no minimum is located by linear search technique.
ERROR='3' means error in gradient calculation.

Method:

FMFP uses a method of conjugate directions, proposed by Davidon. For a quadratic function of n variables the minimum is located within n iterations.

For reference see

R. Fletcher and M. J. Powell, "A Rapidly Convergent Descent Method for Minimization", Computer Journal, vol. 6, iss. 2, 1963, pp. 163-168.

Mathematical Background:

It is assumed that the function f of the n variables $x_1, \ldots, x_n$ (abbreviated as argument vector x) may be computed together with its gradient vector g(x) for any point x. The generalized Taylor expansion for functions of several variables is

$$f(x+u) = f(x) + g(x) \cdot u + \frac{1}{2} u^T G(x)u + \text{higher terms}$$

where g is the gradient vector and G the matrix of second order partial derivatives. Vectors are assumed to be column vectors; u^T means transpose of vector u. It is assumed that in the neighborhood of the required minimum x_{min} the function is approximated closely by the first three terms of its Taylor expansion, giving

$$f(x) = f(x_{min}) + \frac{1}{2}(x - x_{min})^T G(x_{min})(x-x_{min})$$

since $g(x_{min}) = 0$. Then the gradient is seen to be approximately $g(x) = G(x_{min})\ (x - x_{min})$.

Assume now that the symmetric matrix G is positive definite. Then the following equation holds true:

$$x - x_{min} = G^{-1}\ (x_{min}) \cdot g(x)$$

which would allow x_{min} to be calculated in one step if $G^{-1}\ (x_{min})$ were available.

To approach $G^{-1}\ (x_{min})$, a method of successive linear searches in G-conjugate directions is used. Starting with the identity matrix $G^{(0)} = I$, a sequence of symmetric matrices $G^{(i)}$ is generated that approximates G^{-1}. At the (i+1)st iteration step a linear search is made in direction $h^{(i)} = -G^{(i)}g^{(i)}$, where $g^{(i)}$ is an abbreviation for $g(x^{(i)})$. By means of the linear search the minimum of $y(t) = f(x^{(i)} + t \cdot h^{(i)})$ is determined, giving argument $x^{(i+1)} = x^{(i)} + t_i \cdot h^{(i)}$.

The argument of the minimum $x^{(i+1)}$ on the line through $x^{(i)}$ in direction $h^{(i)}$ is determined by the relation that scalar product $(g^{(i+1)}, h^{(i)}) = 0$.

Now:

$$x^{(n)} = x^{(j)} + \sum_{i=j}^{n-1} t_i\ h^{(i)}$$

and:

$$g^{(n)} = g^{(j)} + \sum_{i=j}^{n-1} t_i\ Gh^{(i)}$$

Therefore:

scalar product

$$(g^{(n)}, h^{(j)}) = \sum_{i=j+1}^{n-1} t_i\ (Gh^{(i)}, h^{(j)})$$

Suppose now that the vectors $h^{(0)}, h^{(1)}, \ldots, h^{(n-1)}$ are G-conjugate, satisfying $(Gh^{(i)}, h^{(j)}) = 0$ for $i \neq j$. Then $(g^{(n)}, h^{(j)}) = 0$, and since $h^{(0)}, h^{(1)}, \ldots, h^{(n-1)}$ form a basis, $g^{(n)} = 0$ and $x^{(n)} = x_{min}$. This shows that the minimum is located at the nth iteration for a quadratic function when using successive linear searches for G-conjugate directions.

Programming Considerations:

For the generation of G-conjugate directions, start with $h^{(0)} = -g^{(0)}$ and calculate successive directions $h^{(i)}$ by means of $h^{(i)} = -G^{(i)}g^{(i)}$, where $G^{(i)}$ is modified to $G^{(i+1)}$ so that $h^{(i)}$ is an eigenvector of the matrix $G^{(i+1)}\ G$ with eigenvalue 1. This ensures that $G^{(i)}$ approaches G^{-1} as $x^{(i)}$ approaches x_{min}. An easy calculation shows:

$$G^{(i+1)} = G^{(i)} + \frac{dx \cdot dx^T}{dx^T \cdot dg} - \frac{G^{(i)}dg \cdot dg^T G^{(i)}}{dg^T\ G^{(i)}dg}$$

with $dg = g^{(i+1)} - g^{(i)}$

$dx = x^{(i+1)} - x^{(i)}$

where all vectors are regarded as column vectors, and superscript T means transpose of column vector--that is, row vector.

The strategy adopted for termination of the successive linear searches is as follows:

1. If the function value has not decreased in the last iteration step, the search for the minimum is terminated provided the gradient is already sufficiently small; otherwise, the next step is in the direction of steepest descent.
2. If the argument vector and the direction vector change by very small amounts, and at least n iterations are performed, the minimization is terminated again.
3. If the number of iterations exceeds an upper bound furnished by the user, further calculation is bypassed, and an error code is set to 1 indicating poor convergence.

4. If one of the successive linear searches indicates that no constrained minimum exists, further

calculation is bypassed again and the error code is set to 2, indicating that it is likely that no minimum exists.

The i^{th} term $G^{(i)}$ is reset to the identity matrix if there is indication that the current $G^{(i)}$ is not positive definite, or if the formula for $G^{(i+1)}$ breaks down because of zero divisors.

The linear search technique used in procedure FMFP is as follows:
For a given argument vector x and a vector h defining a direction through x, a local minimum of the function $y(t) = f(x+t \cdot h)$ must be found. This means that a value t_m must be determined for which

$$y'(t_m) = \text{scalar product } (g(x+t_m \cdot h), h) = 0$$

From $y'(0) = (g(x),h) < 0$ it is evident that a minimum $y(t_m) < y(0)$ should be found for positive values of t.

The calculation of the minimum is in three stages:

1. Estimating the magnitude of t_m.
2. Determining an interval containing t_m.
3. Interpolating the value of t_m.

An estimate of the stepsize may be obtained, assuming that the true value of the constrained minimum is equal to the estimated value EST of the unconstrained minimum and that y(t) is closely represented by a quadratic polynomial passing through x, y(0) with derivative y' (0):

$$\text{step} = 2\ (\text{EST} - y(0))/y'(0)$$

This equation tends to overestimate the stepsize since the unconstrained minimum will normally not lie on the line through x with direction h. Therefore step is taken as stepsize s only if it is positive and less than one. Otherwise s = 1 is taken as stepsize.

At the second stage y(t) and y'(t) are examined at the points

$$t = s,\ 2s,\ 4s,\ \dots,\ s_1,\ s_2$$

where successive values are obtained by doubling the stepsize.

This search is terminated at $t = s_2$ if:

$$y'(s_2) = 0$$

or $y'(s_2) > 0$

or $y(s_2) \geq y(s_1)$

or if $s_2 \cdot \left(\sum_{i=1}^{N} \left| h_i \right| \right) > 10^{10}$.

The last case (search argument runs out of range) is interpreted as an indication that no local minimum exists on the given line. Therefore, the error indicator is set to '2' and further calculation is bypassed; that is, FMFP returns the current minimal value with ERROR = '2'.

In case $y'(s_2) = 0, t_m$ is set to s_2 and $x_m = x+s_2 \cdot h$ is used as argument of a constrained minimum on the line through x with direction h.

In the second and third case $y'(s_2) > 0$ and/or $y(s_2) \geq y(s_1)$, a minimum lies necessarily between s_1 and s_2. Its argument value gets approximated using cubic interpolation.

The extrema of the cubic interpolation passing through (s_1, $y_1 = y(s_1)$, $y'_1 = y'(s_1)$) and s_2, $y_2 = y(s_2)$, $y'_2 = y'(s_2)$) are the roots of the quadratic equation

$$y'_1 - 2(z+y'_1)\frac{t-s_1}{s_2-s_1} + (y'_1+y'_2+2z)\left(\frac{t-s_1}{s_2-s_1}\right)^2 = 0$$

with $z = y'_1 + y'_2 - 3\dfrac{y_2-y_1}{s_2-s_1}$

The substitution $\dfrac{t-s_1}{s_2-s_1} = 1-\alpha$ gives

$$y'_2 - 2\alpha(y'_2+z)+\alpha^2(y'_1+y'_2+2z) = 0$$

with the solutions

$$\alpha = \frac{y'_2 + z \pm w}{y'_1+y'_2+2z}$$

where

$$w = +\sqrt{z^2 - y'_1 y'_2}$$

It is interesting to note that $y'_1 < 0$, $y'_2 \geq 0$, as well as $y'_1 < 0$, $y'_2 < 0$, $y_2 \geq y_1$, that is, $|z| < |y'_1 + y'_2|$, guarantee a real value of w. This means the cubic interpolation polynomial has real extrema.

The cubic interpolation polynomial may degenerate to a quadratic if $y'_1 + y'_2 + 2z = 0$

with minimum at

$$\alpha = \frac{y'_2}{y'_2 - y'_1}$$

The sign of w must be so chosen that α belongs to the minimum, which is necessarily between s_1 and s_2.

From

$$\alpha = \frac{y'_2 - z + w}{y'_2 - y'_1 + 2w}$$

$$= \frac{(y'_2 - z + w)\,(y'_2 - y'_1 - 2w)}{(y'_2 - y'_1 + 2w)\,(y'_2 - y'_1 - 2w)}$$

$$= \frac{(y'_2 + z - w)\,(y'_1 + y'_2 - 2z)}{(y'_1 + y'_2 + 2z)(y'_1 + y'_2 - 2z)} = \frac{y'_2 + z - w}{y'_1 + y'_2 + 2z}$$

It is easily seen that

$$\alpha = \frac{y'_2 - z + w}{y'_2 - y'_1 + 2w}$$

respectively, if $y'_2 - y'_1 + 2w = 0$

$$\alpha = \frac{y'_2 + z - w}{y'_1 + y'_2 + 2z}$$

give the argument of the minimum in all cases. The first formula gives extra numerical stability if y'_1 is close to $-y'_2$ and y_1 is close to y_2 and also contains the degenerate case as special. The second formula may be necessary if both extreme values lie between s_1 and s_2. Then the one closer to s_1 is the minimum. (This follows easily from geometrical considerations.)

The following analysis shows that $0 < \alpha < 1$:

$y'_2 > 0,\ y'_1 < 0$ implies $w > |z|$. Hence

$$0 < \frac{y'_2}{y'_2 + 2w - y'_1} < \alpha = \frac{y'_2 + w - z}{y'_2 - y'_1 + 2w} < \frac{y'_2 + 2w}{y'_2 + 2w - y'_1} < 1 \qquad (1)$$

$y_2 \geq y'_1, y'_2 < 0$ implies $0 > z \leq y'_2 + y'_1$ and $w < |z|$.

Hence

$$0 < \frac{-y'_2 - z}{-y'_2 - 2z - y'_1} < \alpha = \frac{y'_2 + z - w}{y'_1 + y'_2 + 2z} < \frac{-y'_2 - 2z}{-y'_2 - 2z - y'_1} < 1 \, . \qquad (2)$$

Note that for the other root

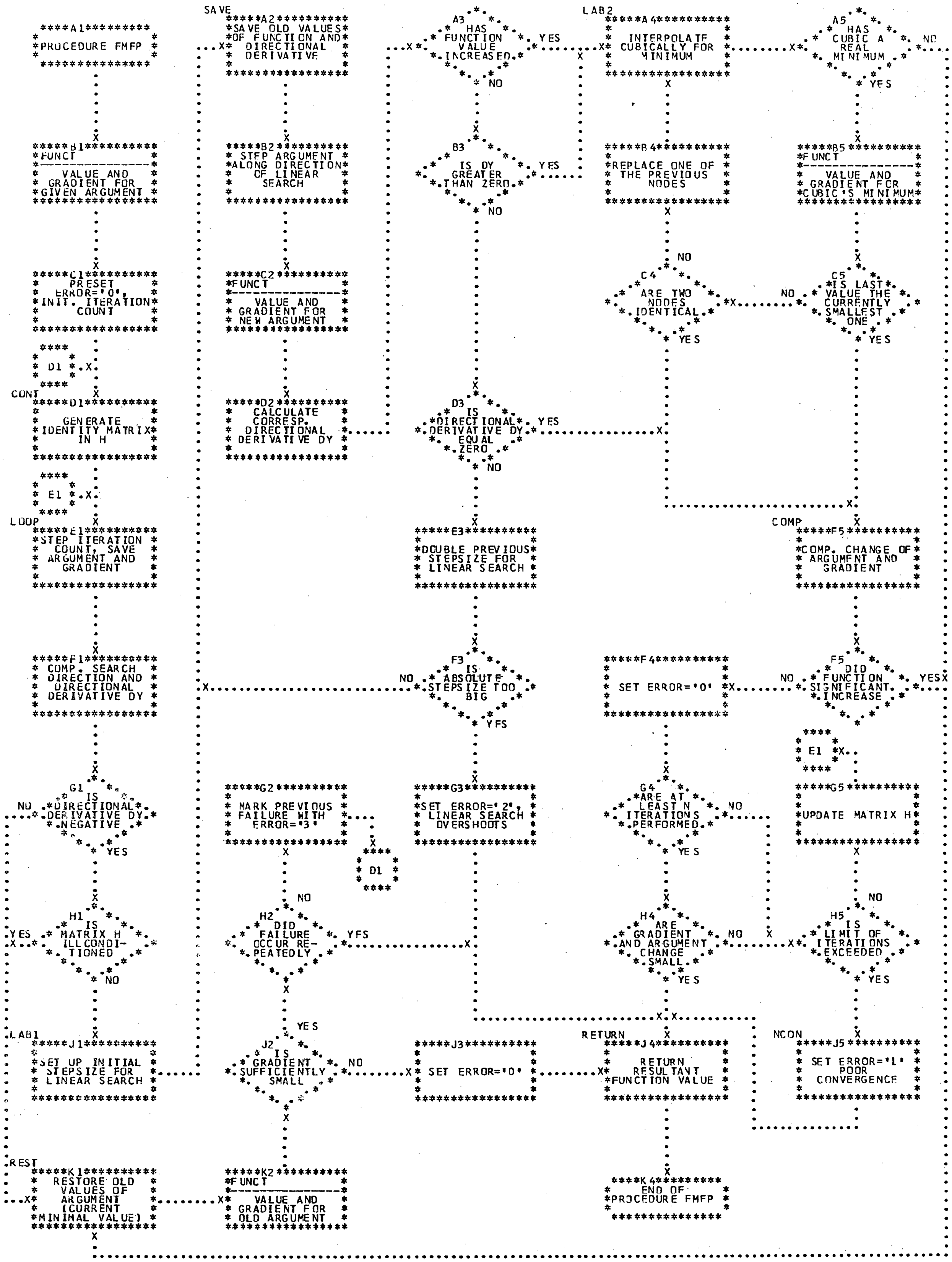
$$\frac{y'_2 + z + w}{y'_1 + y'_2 + 2z} < \frac{-y'_2 - z}{-y'_2 - 2z - y'_1} < \alpha \, .$$

The minimum of the cubic interpolation polynomial is located at

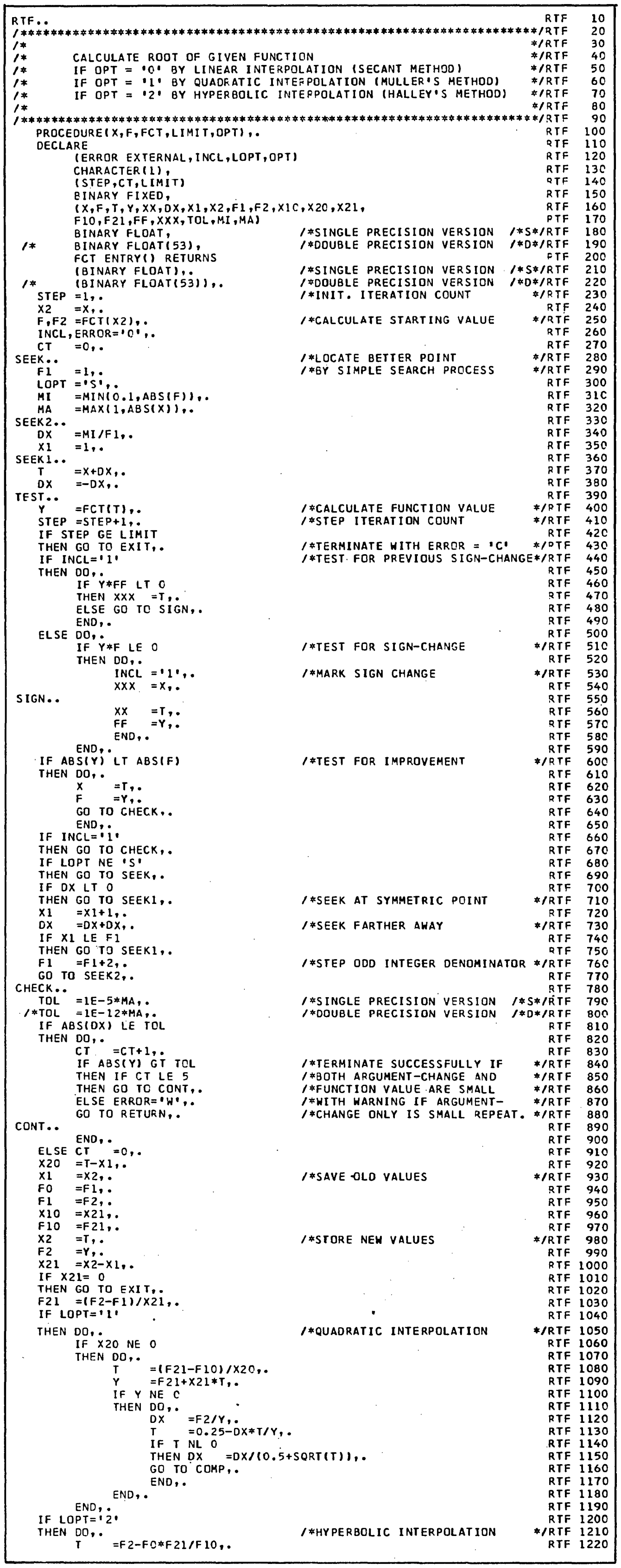
$$s_3 = s_1 + (1-\alpha)\,(s_2 - s_1) = s_2 - \alpha\,(s_2 - s_1)$$

If $y(s_3) \leq y(s_1)$ and $y(s_3) \leq y(s_2)$, then t_m is set equal to s_3 and $x_m = x + t_m \cdot h$ is used as argument of the wanted minimum along the given line. Otherwise, the interval (s_1, s_2) is reduced by replacing s_1 by s_3 if $y(s_3) \leq y(s_1)$ and $y'(s_3) < 0$ and by replacing s_2 by s_3 in all other cases. Then the interpolation process is repeated for this new reduced interval.

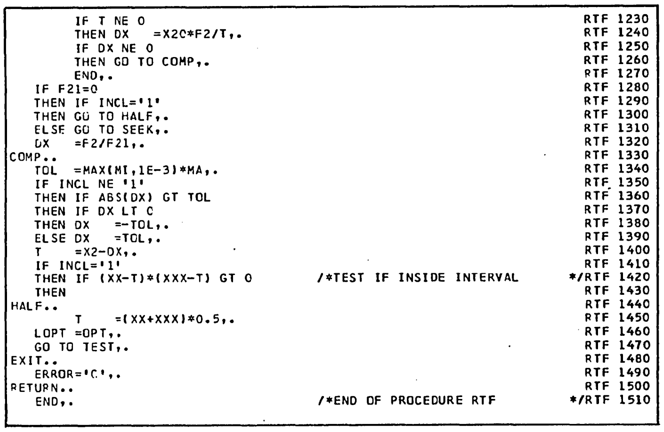
PROCEDURE FMFP DETERMINES AN UNCONSTRAINED MINIMUM OF A FUNCTION OF SEVERAL VARIABLES


A1
PROCEDURE FMFP
SAVE
A2
SAVE OLD VALUES OF FUNCTION AND DIRECTIONAL DERIVATIVE
A3
HAS FUNCTION VALUE INCREASED
YES
NO
LAB2
A4
INTERPOLATE CUBICALLY FOR MINIMUM
A5
HAS CUBIC A REAL MINIMUM
NO
YES
B1
FUNCT
VALUE AND GRADIENT FOR GIVEN ARGUMENT
B2
STEP ARGUMENT ALONG DIRECTION OF LINEAR SEARCH
B3
IS DY GREATER THAN ZERO
YES
NO
B4
REPLACE ONE OF THE PREVIOUS NODES
B5
FUNCT
VALUE AND GRADIENT FOR CUBIC'S MINIMUM
C1
PRESET ERROR='0', INIT. ITERATION COUNT
C2
FUNCT
VALUE AND GRADIENT FOR NEW ARGUMENT
C4
ARE TWO NODES IDENTICAL
NO
YES
C5
IS LAST VALUE THE CURRENTLY SMALLEST ONE
NO
YES
D1
CONT
D1
GENERATE IDENTITY MATRIX IN H
D2
CALCULATE CORRESP. DIRECTIONAL DERIVATIVE DY
D3
IS DIRECTIONAL DERIVATIVE DY EQUAL ZERO
YES
NO
E1
LOOP
E1
STEP ITERATION COUNT, SAVE ARGUMENT AND GRADIENT
E3
DOUBLE PREVIOUS STEPSIZE FOR LINEAR SEARCH
COMP
F5
COMP. CHANGE OF ARGUMENT AND GRADIENT
F1
COMP. SEARCH DIRECTION AND DIRECTIONAL DERIVATIVE DY
F3
IS ABSOLUTE STEPSIZE TOO BIG
NO
YES
F4
SET ERROR='0'
F5
DID FUNCTION SIGNIFICANT. INCREASE
NO
YES
E1
G1
IS DIRECTIONAL DERIVATIVE DY. NEGATIVE
NO
YES
G2
MARK PREVIOUS FAILURE WITH ERROR='3'
D1
G3
SET ERROR='2', LINEAR SEARCH OVERSHOOTS
G4
ARE AT LEAST N ITERATIONS PERFORMED
NO
YES
G5
UPDATE MATRIX H
H1
IS MATRIX H ILLCONDITIONED
YES
NO
H2
DID FAILURE OCCUR REPEATEDLY
NO
YES
H4
ARE GRADIENT AND ARGUMENT CHANGE SMALL
NO
YES
H5
IS LIMIT OF ITERATIONS EXCEEDED
NO
YES
LAB1
J1
SET UP INITIAL STEPSIZE FOR LINEAR SEARCH
J2
IS GRADIENT SUFFICIENTLY SMALL
YES
NO
J3
SET ERROR='0'
RETURN
J4
RETURN RESULTANT FUNCTION VALUE
NCON
J5
SET ERROR='1' POOR CONVERGENCE
REST
K1
RESTORE OLD VALUES OF ARGUMENT (CURRENT MINIMAL VALUE)
K2
FUNCT
VALUE AND GRADIENT FOR OLD ARGUMENT
K4
END OF PROCEDURE FMFP

- Subroutine RTF



```
RTF..
/**********************************************************************/
/*                                                                    */
/*      CALCULATE ROOT OF GIVEN FUNCTION                              */
/*      IF OPT = '0' BY LINEAR INTERPOLATION (SECANT METHOD)          */
/*      IF OPT = '1' BY QUADRATIC INTERPOLATION (MULLER'S METHOD)     */
/*      IF OPT = '2' BY HYPERBOLIC INTERPOLATION (HALLEY'S METHOD)    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,F,FCT,LIMIT,OPT),.
   DECLARE
        (ERROR EXTERNAL,INCL,LOPT,OPT)
        CHARACTER(1),
        (STEP,CT,LIMIT)
        BINARY FIXED,
        (X,F,T,Y,XX,DX,X1,X2,F1,F2,X1C,X20,X21,
        F10,F21,FF,XXX,TOL,MI,MA)
        BINARY FLOAT,                /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),            /*DOUBLE PRECISION VERSION  /*D*/
        FCT ENTRY() RETURNS
        (BINARY FLOAT),.             /*SINGLE PRECISION VERSION  /*S*/
 /*     (BINARY FLOAT(53)),.         /*DOUBLE PRECISION VERSION  /*D*/
   STEP =1,.                         /*INIT. ITERATION COUNT        */
   X2   =X,.
   F,F2 =FCT(X2),.                   /*CALCULATE STARTING VALUE     */
   INCL,ERROR='0',.
   CT   =0,.
SEEK..                               /*LOCATE BETTER POINT          */
   F1   =1,.                         /*BY SIMPLE SEARCH PROCESS     */
   LOPT ='S',.
   MI   =MIN(0.1,ABS(F)),.
   MA   =MAX(1,ABS(X)),.
SEEK2..
   DX   =MI/F1,.
   X1   =1,.
SEEK1..
   T    =X+DX,.
   DX   =-DX,.
TEST..
   Y    =FCT(T),.                    /*CALCULATE FUNCTION VALUE     */
   STEP =STEP+1,.                    /*STEP ITERATION COUNT         */
   IF STEP GE LIMIT
   THEN GO TO EXIT,.                 /*TERMINATE WITH ERROR = 'C'   */
   IF INCL='1'                       /*TEST FOR PREVIOUS SIGN-CHANGE*/
   THEN DO,.
        IF Y*FF LT 0
        THEN XXX  =T,.
        ELSE GO TO SIGN,.
        END,.
   ELSE DO,.
        IF Y*F LE 0                  /*TEST FOR SIGN-CHANGE         */
        THEN DO,.
             INCL ='1',.             /*MARK SIGN CHANGE             */
             XXX  =X,.
SIGN..
             XX   =T,.
             FF   =Y,.
             END,.
        END,.
   IF ABS(Y) LT ABS(F)               /*TEST FOR IMPROVEMENT         */
   THEN DO,.
        X    =T,.
        F    =Y,.
        GO TO CHECK,.
        END,.
   IF INCL='1'
   THEN GO TO CHECK,.
   IF LOPT NE 'S'
   THEN GO TO SEEK,.
   IF DX LT 0
   THEN GO TO SEEK1,.                /*SEEK AT SYMMETRIC POINT      */
   X1   =X1+1,.
   DX   =DX+DX,.                     /*SEEK FARTHER AWAY            */
   IF X1 LE F1
   THEN GO TO SEEK1,.
   F1   =F1+2,.                      /*STEP ODD INTEGER DENOMINATOR */
   GO TO SEEK2,.
CHECK..
   TOL  =1E-5*MA,.                   /*SINGLE PRECISION VERSION  /*S*/
 /*TOL  =1E-12*MA,.                  /*DOUBLE PRECISION VERSION  /*D*/
   IF ABS(DX) LE TOL
   THEN DO,.
        CT   =CT+1,.
        IF ABS(Y) GT TOL             /*TERMINATE SUCCESSFULLY IF    */
        THEN IF CT LE 5              /*BOTH ARGUMENT-CHANGE AND     */
        THEN GO TO CONT,.            /*FUNCTION VALUE ARE SMALL     */
        ELSE ERROR='W',.             /*WITH WARNING IF ARGUMENT-    */
        GO TO RETURN,.               /*CHANGE ONLY IS SMALL REPEAT. */
CONT..
        END,.
   ELSE CT   =0,.
   X20  =T-X1,.
   X1   =X2,.                        /*SAVE OLD VALUES              */
   F0   =F1,.
   F1   =F2,.
   X10  =X21,.
   F10  =F21,.
   X2   =T,.                         /*STORE NEW VALUES             */
   F2   =Y,.
   X21  =X2-X1,.
   IF X21= 0
   THEN GO TO EXIT,.
   F21  =(F2-F1)/X21,.
   IF LOPT='1'
   THEN DO,.                         /*QUADRATIC INTERPOLATION      */
        IF X20 NE 0
        THEN DO,.
             T    =(F21-F10)/X20,.
             Y    =F21+X21*T,.
             IF Y NE C
             THEN DO,.
                  DX   =F2/Y,.
                  T    =0.25-DX*T/Y,.
                  IF T NL 0
                  THEN DX   =DX/(0.5+SQRT(T)),.
                  GO TO COMP,.
                  END,.
             END,.
        END,.
   IF LOPT='2'
   THEN DO,.                         /*HYPERBOLIC INTERPOLATION     */
        T    =F2-FC*F21/F10,.
```



```
        IF T NE 0
        THEN DX   =X2C*F2/T,.
        IF DX NE 0
        THEN GO TO COMP,.
        END,.
   IF F21=C
   THEN IF INCL='1'
   THEN GO TO HALF,.
   ELSE GO TO SEEK,.
   DX   =F2/F21,.
COMP..
   TOL  =MAX(MI,1E-3)*MA,.
   IF INCL NE '1'
   THEN IF ABS(DX) GT TOL
   THEN IF DX LT C
   THEN DX   =-TOL,.
   ELSE DX   =TOL,.
   T    =X2-DX,.
   IF INCL='1'
   THEN IF (XX-T)*(XXX-T) GT 0       /*TEST IF INSIDE INTERVAL      */
   THEN
HALF..
        T    =(XX+XXX)*0.5,.
   LOPT =OPT,.
   GO TO TEST,.
EXIT..
   ERROR='C',.
RETURN..
   END,.                             /*END OF PROCEDURE RTF         */
```

Purpose:

RTF refines a given initial guess for a root of the general (transcendental) equation f(x) = 0 using:

linear interpolation if OPT='0' (secant method)
quadratic interpolation if OPT='1'
hyperbolic interpolation if OPT='2'

Usage:

CALL RTF (X, F, FCT, LIMIT, OPT);

X - BINARY FLOAT [(53)]
Given initial guess for root of f(x) = 0.
Resultant refined approximation for root of f(x) = 0.

F - BINARY FLOAT [(53)]
Resultant function value for calculated value of x.

FCT - ENTRY (BINARY FLOAT [(53)]) RETURNS (BINARY FLOAT [(53)])
Given function procedure for calculation of the function values f(x). It must be supplied by the user.

Usage:
FCT(T)
FCT(T) - BINARY FLOAT [(53)]
Resultant function value f(t).
T - BINARY FLOAT [(53)]
Given argument of function.

LIMIT - BINARY FIXED
Given bound for the number of function evaluations to be performed at most.

OPT - CHARACTER(1)
Given option for selection of iteration method.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR='C' means no convergence obtained within LIMIT function evaluations, possibly because of poor initial guess or unrealistically small value of LIMIT.

ERROR='W' means small changes in successive refined approximations indicate covergence of method, while corresponding function values are not small enough. Possibly the function values cannot be obtained accurately enough by the user-supplied procedure FCT. The returned value of x has the absolutely smallest function value f(x) among all arguments used in the course of calculation.

Any value of OPT different from '1' and '2' is treated as if it were '0'.

Method:

A refined approximation of the root is calculated as root of the linear fit through two successive approximations if OPT='0' (secant method).

The root of a quadratic fit through three successive approximations is used if OPT='1' (Muller's method).

With OPT='2' the refined approximation is calculated as root of a hyperbolic fit through three successive approximations.

For reference see:

J. F. Traub, "The Solution of Transcendental Equations", edited by A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, vol. 2, pp. 171-184.

Mathematical Background:

Secant iteration method

The linear interpolation polynomial through two successive approximants is given by (Newtonian formulation)

$$P(t) = f(x_i) + f\left[x_i, x_{i-1}\right](t-x_i),$$

where

$$f\left[x_i, x_{i-1}\right] = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \tag{1}$$

A refined approximation is obtained setting $P(x_{i+1}) = 0$:

$$x_{i+1} = x_i - f(x_i)/f\left[x_i, x_{i-1}\right], \text{ for } i \geq 2$$

and

$$f(x_i) \neq f(x_{i-1}) \tag{2}$$

The asymptotic order of convergence is p = 1.62.

Muller's iteration method

The quadratic interpolation polynomial through three successive approximants is given by

$$P(t) = f(x_i) + f\left[x_i, x_{i-1}\right](t-x_i) + f\left[x_i, x_{i-1}, x_{i-2}\right](t-x_i)(t-x_{i-1}) \tag{3}$$

With the notation

$$2w = f\left[x_i, x_{i-1}\right] + f\left[x_i, x_{i-1}, x_{i-2}\right](x_i-x_{i-1}) \tag{4}$$

this reads

$$P(t) = f(x_i) + 2w\,(t-x_i) + f\left[x_i, x_{i-1}, x_{i-2}\right](t-x_i)^2 \tag{5}$$

A refined approximation is obtained setting $P(x_{i+1}) = 0$:

$$x_{i+1} = x_i - w\left(1 - \sqrt{1 - f(x_i)f\left[x_i, x_{i-1}, x_{i-2}\right]/w^2}\right) / f\left[x_i, x_{i-1}, x_{i-2}\right]$$

or preferably

$$x_{i+1} = x_i - \frac{f(x_i)}{w\left(1 + \sqrt{1 - f(x_i)f\left[x_i, x_{i-1}, x_{i-2}\right]/w^2}\right)} \tag{6}$$

with $w \neq 0$ and $f(x_i) \cdot f\left[x_i, x_{i-1}, x_{i-2}\right] \leq w^2$

The asymptotic order of convergence is p = 1.84.

Hyperbolic interpolation iteration method

Hyperbolic interpolation is defined through

$$P(t) = (t-a) / (b+ct)$$

with

$$(b+c\,x_j)\, f(x_j) = x_j - a, \text{ for } j = i,\ i-1,\ i-2. \qquad (7)$$

A refined approximation is obtained setting $P(x_{i+1}) = 0$, that is, $x_{i+1} = a$.

Symmetric formula:

$$x_{i+1} = \frac{x_i \cdot (x_{i-2}-x_{i-1})/f(x_i) + x_{i-1} \cdot (x_i-x_{i-2})/f(x_{i-1}) + x_{i-2}(x_{i-1}-x_i)/f(x_{i-2})}{(x_{i-2}-x_{i-1})/f(x_i) + (x_i-x_{i-2})/f(x_{i-1}) + (x_{i-1}-x_i)/f(x_{i-2})} \qquad (8)$$

x_{i+1} is a weighted mean of x_i, x_{i-1}, x_{i-2}.

Preferable is the equivalent unsymmetric formula:

$$x_{i+1} = x_i - \cfrac{x_i - x_{i-2}}{1 - \cfrac{f(x_{i-2}) \cdot f\,[x_i,\, x_{i-1}\,]}{f(x_i) \cdot f\,[x_{i-1},\, x_{i-2}]}} \qquad (9)$$

with

$$f(x_{i-2}) \cdot f\,[x_i,\, x_{i-1}] \neq f(x_i) \cdot f\,[x_{i-1},\, x_{i-2}] \neq 0$$

The asymptotic order of convergence is $p = 1.84$.

Programming Considerations:

1. The three above-defined iteration methods (1), (6), and (9) are combined with a search method that uses arguments

$$x \pm 2^k \cdot \Delta/(2i+1) \quad \text{for} \begin{cases} i = 0, 1, \ldots, k \\ k = 0, 1, \ldots \end{cases} \qquad (10)$$

until an argument t is found for which either

$$|f(t)| < |f(x)| \quad \text{or } f(t) \cdot f(x) \leq 0.$$

The value of Δ used internally is $\Delta = \min(0.1, |f(x)|)$.

2. If an interval (x_l, x_u) enclosing a root has been found, that is, $f(x_l) \cdot f(x_u) < 0$, then successive approximants from one of the iteration methods above must lie inside this interval. Otherwise, $(x_l+x_u)/2$ is used as next approximation. The interval bounds for this bisection method are updated in the course of calculation.

3. If no sign change has been located previously, the absolute argument change at a single iteration step is reduced to $\max(0.001, \Delta) \cdot \max(1, |X|)$ if necessary, in order to avoid overshooting and overflow problems.

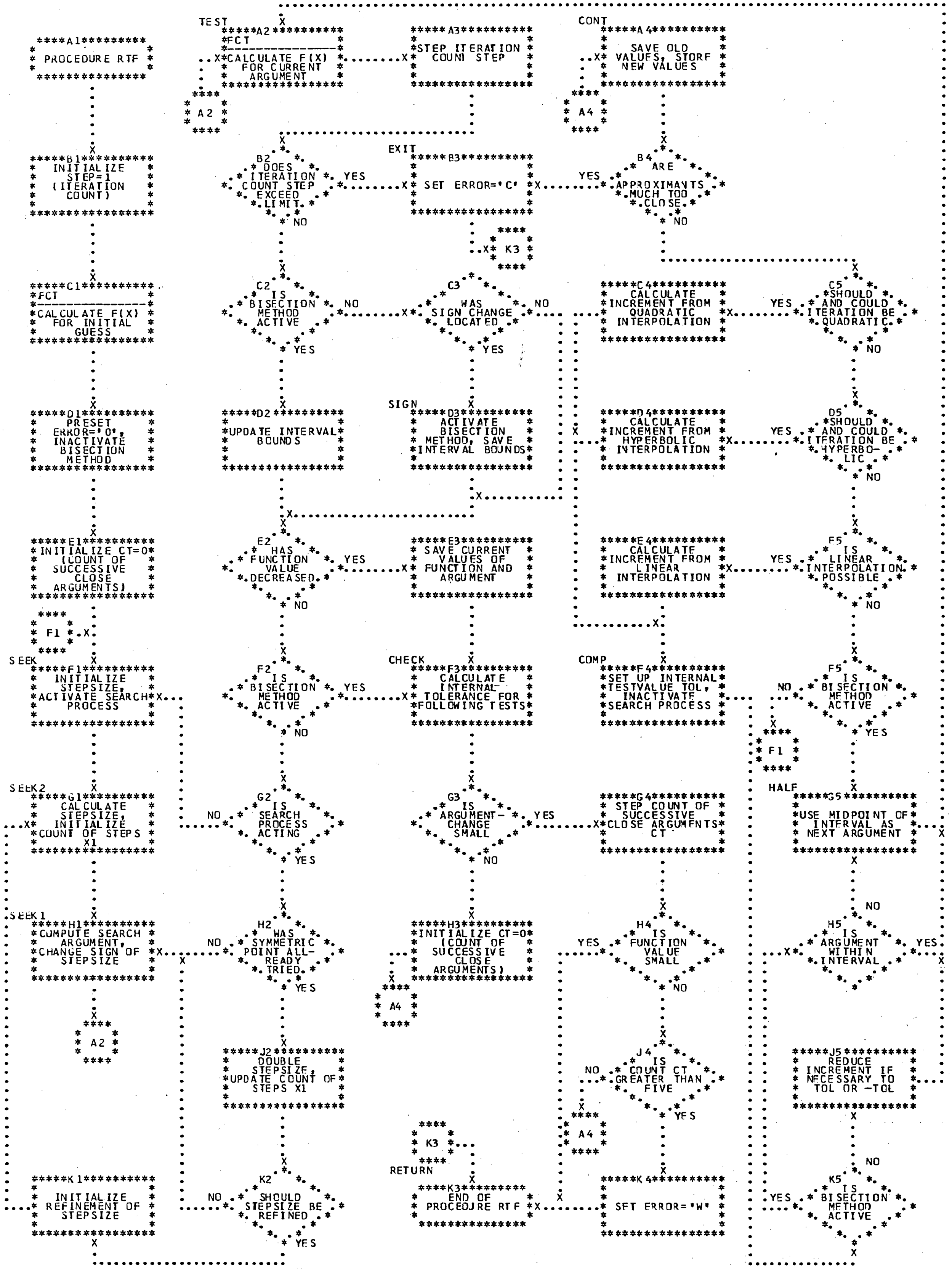
4. If, in case of no previous sign change, the iteration method fails to give an argument x_{i+1} for which either $f(x_{i+1}) \cdot f(x_i) \leq 0$ or $|f(x_{i+1})| < |f(x_i)|$, then the next approximant is calculated by the search method (1).

5. Calculation of the first approximant is based on the simple search method, while the second approximant is calculated with the secant method.

6. The convergence test used requires that both argument change and function value are absolutely less or equal to $10^{-5} \cdot \max(1, |X|)$ in single precision and $10^{-12} \cdot \max(1, |X|)$ in double precision. If the argument change is absolutely less than or equal to this internal tolerance five times in sequence, while the function values are not small enough, then the currently best values x, f(x) are returned with ERROW='W'.

7. The iteration process is terminated with ERROR='C' if the number of function evaluations exceeds the user-specified limit LIMIT.

PROCEDURE RTF REFINES AN INITIAL GUESS FOR A ROOT OF F(X)=0 USING LINEAR, QUADRATIC OR HYPERBOLIC INTERPOLATION



A1 PROCEDURE RTF
TEST
A2 FCT CALCULATE F(X) FOR CURRENT ARGUMENT
A3 STEP ITERATION COUNT STEP
CONT
A4 SAVE OLD VALUES, STORE NEW VALUES
A2
A4
B1 INITIALIZE STEP=1 (ITERATION COUNT)
B2 DOES ITERATION COUNT STEP EXCEED LIMIT
YES
NO
EXIT
B3 SET ERROR='C'
K3
B4 ARE APPROXIMANTS MUCH TOO CLOSE
YES
NO
C1 FCT CALCULATE F(X) FOR INITIAL GUESS
C2 IS BISECTION METHOD ACTIVE
NO
YES
C3 WAS SIGN CHANGE LOCATED
NO
YES
C4 CALCULATE INCREMENT FROM QUADRATIC INTERPOLATION
C5 SHOULD AND COULD ITERATION BE QUADRATIC
YES
NO
D1 PRESET ERROR='0', INACTIVATE BISECTION METHOD
D2 UPDATE INTERVAL BOUNDS
SIGN
D3 ACTIVATE BISECTION METHOD, SAVE INTERVAL BOUNDS
D4 CALCULATE INCREMENT FROM HYPERBOLIC INTERPOLATION
D5 SHOULD AND COULD ITERATION BE HYPERBOLIC
YES
NO
E1 INITIALIZE CT=0 (COUNT OF SUCCESSIVE CLOSE ARGUMENTS)
E2 HAS FUNCTION VALUE DECREASED
YES
NO
E3 SAVE CURRENT VALUES OF FUNCTION AND ARGUMENT
E4 CALCULATE INCREMENT FROM LINEAR INTERPOLATION
E5 IS LINEAR INTERPOLATION POSSIBLE
YES
NO
F1
SEEK
F1 INITIALIZE STEPSIZE, ACTIVATE SEARCH PROCESS
F2 IS BISECTION METHOD ACTIVE
YES
NO
CHECK
F3 CALCULATE INTERNAL TOLERANCE FOR FOLLOWING TESTS
COMP
F4 SET UP INTERNAL TESTVALUE TOL, INACTIVATE SEARCH PROCESS
F5 IS BISECTION METHOD ACTIVE
NO
YES
F1
SEEK2
G1 CALCULATE STEPSIZE, INITIALIZE COUNT OF STEPS X1
G2 IS SEARCH PROCESS ACTING
NO
YES
G3 IS ARGUMENT-CHANGE SMALL
YES
NO
G4 STEP COUNT OF SUCCESSIVE CLOSE ARGUMENTS CT
HALF
G5 USE MIDPOINT OF INTERVAL AS NEXT ARGUMENT
SEEK1
H1 COMPUTE SEARCH ARGUMENT, CHANGE SIGN OF STEPSIZE
H2 WAS SYMMETRIC POINT ALL-READY TRIED
NO
YES
H3 INITIALIZE CT=0 (COUNT OF SUCCESSIVE CLOSE ARGUMENTS)
A4
H4 IS FUNCTION VALUE SMALL
YES
NO
H5 IS ARGUMENT WITHIN INTERVAL
YES
NO
A2
J2 DOUBLE STEPSIZE, UPDATE COUNT OF STEPS X1
J4 IS COUNT CT GREATER THAN FIVE
NO
YES
A4
J5 REDUCE INCREMENT IF NECESSARY TO TOL OR -TOL
K3
RETURN
K1 INITIALIZE REFINEMENT OF STEPSIZE
K2 SHOULD STEPSIZE BE REFINED
NO
YES
K3 END OF PROCEDURE RTF
K4 SET ERROR='W'
K5 IS BISECTION METHOD ACTIVE
YES
NO

- <u>Subroutine</u> RTFD

```
RTFD..
/*******************************************************************/
/*                                                                 */
/*      CALCULATE ROOT OF GIVEN FUNCTION USING DERIVATIVE VALUES.  */
/*      IF OPT = 'C' BY LINEAR INTERPOLATION (NEWTON METHOD)       */
/*      IF OPT = '1' BY INVERSE QUADPATIC INTERPOLATION            */
/*      IF OPT = '2' BY HYPERBOLIC INTERPOLATION (HALLEY METHOD)   */
/*                                                                 */
/*******************************************************************/
   PROCEDURE(X,F,DF,FCT,LIMIT,OPT),.
   DECLAPE
        (ERROR EXTERNAL,INCL,LOPT,OPT)
        CHARACTER(1),
        (STEP,CT,LIMIT)
        BINARY FIXED,
        (X,F,T,Y,XX,DX,X1,X2,F1,F2,DF1,DF2,DY,DF,TOL,MI,MA,FF,XXX)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),              /*DOUBLE PRECISION VERSION  /*D*/
        FCT ENTRY,.
   STEP =1,.
   X2   =X,.
   CALL FCT(X2,F2,DF2),.               /*CALCULATE STARTING VALUE     */
   F    =F2,.
   DF   =DF2,.
   INCL,ERROR='C',.
   CT   =0,.
   LOPT ='0',.                         /*NO PREVIOUS VALUE AVAILABLE  */
   GO TO COMP,.                        /*USE NEWTON METHOD            */
SEEK..                                 /*LOCATE BETTER POINT          */
   F1   =1,.                           /*BY SIMPLE SEARCH PROCESS     */
   LOPT ='S',.
SEEK2..
   DX   =MI/F1,.
   X1   =1,.
SEEK1..
   T    =X+DX,.
   DX   =-DX,.
TEST..
   CALL FCT(T,Y,DY),.                  /*CALCULATE FUNCTION VALUE     */
   STEP =STEP+1,.                      /*STEP ITERATION COUNT         */
   IF STEP GE LIMIT
   THEN GO TO EXIT,.                   /*TERMINATE WITH ERROR ='C'    */
   IF INCL='1'                         /*TEST FOR PREVIOUS SIGN-CHANGE*/
   THEN DO,.
        IF Y*FF LT 0
        THEN XXX  =T,.
        ELSE GO TO SIGN,.
        END,.
   ELSE DO,.
        IF Y*F LE 0                    /*TEST FOR SIGN-CHANGE         */
        THEN DO,.
             INCL ='1',.               /*MARK SIGN CHANGE             */
             XXX  =X,.
SIGN..
             XX   =T,.
             FF   =Y,.
             END,.
        END,.
   IF ABS(Y) LT ABS(F)                 /*TEST FOR IMPROVEMENT         */
   THEN DO,.
        X    =T,.
        F    =Y,.
        DF   =DY,.
        GO TO CHECK,.
        END,.
   IF INCL='1'
   THEN GO TO CHECK,.
   IF LOPT NE 'S'
   THEN GO TO SEEK,.
   IF DX LT 0
   THEN GO TO SEEK1,.                  /*SEEK AT SYMMETRIC POINT      */
   X1   =X1+1,.
   DX   =DX+DX,.                       /*SEEK FARTHER AWAY            */
   IF X1 LE F1
   THEN GO TO SEEK1,.
   F1   =F1+2,.                        /*STEP ODD INTEGER DENOMINATOR */
   GO TO SEEK2,.
CHECK..
   TOL  =1E-5*MA,.                     /*SINGLE PRECISION VERSION  /*S*/
 /*TOL  =1E-12*MA,.                    /*DOUBLE PRECISION VERSION  /*D*/
   IF ABS(DX) LE TOL
   THEN DO,.
        CT   =CT+1,.
        IF ABS(Y) GT TOL               /*TERMINATE SUCCESSFULLY IF    */
        THEN IF CT LE 5                /*BOTH ARGUMENT-CHANGE AND     */
        THEN GO TO CONT,.              /*FUNCTION VALUE ARE SMALL     */
        ELSE ERROR='W',.               /*WITH WARNING IF ARGUMENT-    */
        GO TO RETURN,.                 /*CHANGE ONLY IS SMALL REPEAT. */
CONT..
        END,.
   ELSE CT   =0,.
   X1   =X2,.                          /*SAVE OLD VALUES              */
   F1   =F2,.
   DF1  =DF2,.
   X2   =T,.                           /*STORE NEW VALUES             */
   F2   =Y,.
   DF2  =DY,.
   DY   =X2-X1,.
   IF DY= 0
   THEN GO TO EXIT,.
COMP..
   MA   =MAX(1,ABS(X)),.
   MI   =MIN(0.1,ABS(F)),.
   IF DF2 NE 0
   THEN DO,.
        DX   =F2/DF2,.                 /*NEWTON METHOD                */
        IF LOPT NE '0'
        THEN DO,.
             T    =(F2-F1)/DY,.
             Y    =DF2-T,.
             T    =DX*(DF1-T+Y+Y)/(DF2*DY),.
             IF LOPT='1'               /*MODIFICATION..               */
             THEN DX   =DX*(1+T),.     /*INVERSE QUADRATIC INTERPOLAT.*/
             IF LOPT='2'               /*MODIFICATION..               */
             THEN IF T NE 1            /*HYPERBOLIC INTERPOLATION     */
             THEN DX   =DX/(1-T),.
             END,.
        LOPT =OPT,.
        TOL  =MAX(MI,1E-3)*MA,.
        IF INCL NE '1'
        THEN DO,.
             IF ABS(DX) GT TOL
             THEN IF DX LT 0
             THEN DX   =-TOL,.
             ELSE DX   = TOL,.
             END,.
        T    =X2-DX,.
        IF INCL='1'
        THEN IF (XX-T)*(XXX-T) GT 0    /*TEST IF INSIDE INTERVAL      */
        THEN
HALF..
             T    =(XX+XXX)*0.5,.
        GO TO TEST,.
        END,.
   ELSE IF INCL='1'
   THEN GO TO HALF,.
   ELSE GO TO SEEK,.
EXIT..
   ERROR='C',.
RETURN..
   END,.                               /*END OF PROCEDURE RTFD        */
```

Purpose:

RTFD refines a given initial guess for a root of the general (transcendental) equation f(x) = 0 using:
Linear interpolation if OPT='0'(Newton method)
Inverse quadratic interpolation if OPT='1'
Hyperbolic interpolation if OPT='2'

Usage:

CALL RTFD(X, F, DF, FCT, LIMIT, OPT);

X - BINARY FLOAT [(53)]
Given initial guess for root of f(x) = 0.
Resultant refined approximation for root of f(x) = 0.

F - BINARY FLOAT [(53)]
Resultant function value f(x) for returned X.

DF - BINARY FLOAT [(53)]
Resultant value of derivative f'(x) for returned X.

FCT - ENTRY (BINARY FLOAT [(53)], BINARY FLOAT [(53)], BINARY FLOAT [(53)])
Given procedure for calculation of values f(x), f'(x). It must be supplied by the user.

Usage:
CALL FCT(X, F, DF);

X - BINARY FLOAT [(53)]
Given argument value.

F - BINARY FLOAT [(53)]
Resultant function value f(x).

DF - BINARY FLOAT [(53)]
Resultant derivative value f'(x).

LIMIT - BINARY FIXED
Given bound for the number of function evaluations to be performed at most.

OPT - CHARACTER (1)
Given option for selection of iteration method.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR='C' - means no convergence is obtained within LIMIT function evaluations, possibly because of poor initial guess or unrealistic small value of LIMIT.

ERROR='W' - means that small changes in successive approximations indicate covergence of method, while corresponding function values are not small enough. Possibly the function values cannot be obtained accurately enough by the user-supplied procedure FCT.

The returned value of X has the absolutely smallest function value f(x) among all arguments tried during the iteration process.

Any value of OPT different from '1' and '2' is treated as if it were '0'.

Method:

A refined approximation of the root is calculated using Newton's method if OPT='0', higher-order methods doing inverse quadratic interpolation if OPT='1', and hperbolic interpolation if OPT='2'. With the higher-order methods the second derivative is estimated from a cubic interpolation polynomial through two successive approximations.

For reference see:

J. F. Traub, "The Solution of Transcendental Equations", edited by A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, vol. 2, pp. 171 - 184.

Mathematical Background:

Newton's iteration method

The linear interpolation polynomial passing through x_i, $f(x_i)$ with derivative $f'(x_i)$ is given by

$$P(t) = f(x_i) + f'(x_i)\,(t-x_i) \tag{1}$$

A refined approximation is obtained setting $P(x_{i+1}) = 0$:

$$x_{i+1} = x_i - f(x_i)/f'(x_i), \text{ for } i \geq 1 \text{ and } f'(x_i) \neq 0.$$

The asymptotic order of convergence is $p = 2$.

Inverse quadratic interpolation

Let $x = F(y)$ denote the inverse function of $y = f(x)$. The quadratic polynomial $Q(y)$ passing through point y_i, x_i with derivatives $F'(y_i)$, $F''(y_i)$ is given by

$$Q(y) = F(y_i) + F'(y_i)\,(y-y_i) + \frac{F''}{2!}(y_i)\,(y-y_i)^2 \tag{2}$$

A refined approximation is obtained setting $x_{i+1} = Q(0)$:

$$x_{i+1} = F(y_i) - F'(y_i)\,y_i + \frac{F''}{2!}(y_i)\,y_i^2 \tag{3}$$

From the identity $x = F(f(x))$ follows easily:

$$F'(y) = \frac{dF}{dy} = 1 \Big/ \frac{df}{dx} = \frac{1}{f'(x)}$$

$$F''(y) = \frac{d^2F}{dy^2} = -\frac{f''(x)}{(f'(x))^3}$$

Hence

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}\left(1 + \frac{f(x_i)}{f'(x_i)}\,\frac{f''(x_i)}{2f'(x_i)}\right) \tag{4}$$

The asymptotic order of convergence is $p = 3$.

Hyperbolic interpolation (Halley's iteration method)

Hyperbolic interpolation is defined by

$$P(t) = (t-a)/(b+ct) \tag{5}$$

with

$$P(x_i) = f(x_i),\quad P'(x_i) = f'(x_i),\quad P''(x_i) = f''(x_i)$$

A refined approximation is obtained setting $P(x_{i+1}) = 0$, that is, $x_{i+1} = a$.

From

$P(t)\,(b+ct) = t-a$ follows, by differentiation, (6)

$$f(x_i)(b+cx_i) = x_i-a$$
$$f'(x_i)(b+cx_i) = 1-f(x_i)\cdot c$$
$$f''(x_i)(b+cx_i) = -2f'(x_i)\cdot c$$

and from the last two equations

$$b+cx_i = -\frac{2f'(x_i)}{f(x_i)f''(x_i)-2(f'(x_i))^2}$$

and

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i) - \dfrac{f(x_i)\, f''(x_i)}{2f'(x_i)}} \tag{7}$$

The asymptotic order of convergence is $p = 3$.

Estimation of second derivative

A cubic interpolation polynomial passing through points x_i, $f(x_i)$ and x_{i-1}, $f(x_{i-1})$ is of the form

$$P(x) = f(x_i) + (x-x_i)f'(x_i) + \alpha(x-x_i)^2 + \beta(x-x_i)^2(x-x_{i-1}) \tag{8}$$

$P(x_i) = f(x_i)$ and $P'(x_i) = f'(x_i)$ are already satisfied. If we set

$P(x_{i-1}) = f(x_{i-1})$ and $P'(x_{i-1}) = f'(x_{i-1})$ then

$$\alpha = \frac{f[x_i, x_{i-1}] - f'(x_i)}{x_{i-1} - x_i}$$

and

$$\beta = \frac{f'(x_i) + f'(x_{i-1}) - 2f[x_i, x_{i-1}]}{(x_{i-1} - x_i)^2}$$

The second derivative $f''(x_i)$ is estimated by

$$P''(x_i) = 2(\alpha + \beta(x_i-x_{i-1})) = 2\left(2f'(x_i) + \frac{f'(x_{i-1}) - 3f[x_i,x_{i-1}]}{(x_i - x_{i-1})}\right) \tag{9}$$

Derivative estimated iteration methods

Replacing $f''(x_i)$ in (4) and (7) by $P''(x_i)$ gives

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \cdot \left(1 + \frac{f(x_i)}{f'(x_i)} \; \frac{2f'(x_i) + f'(x_{i-1}) - 3f[x_i,x_{i-1}]}{(x_i - x_{i-1})f'(x_i)}\right) \tag{4'}$$

and

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \cdot \frac{1}{1 - \dfrac{f(x_i)}{f'(x_i)} \cdot \dfrac{2f'(x_i) + f'(x_{i-1}) - 3f[x_i, x_{i-1}]}{(x_i-x_{i-1})\, f'(x_i)}} \tag{7'}$$

The asymptotic order of both these iteration methods is $p = 2.73$.

Programming Considerations:

1. The three above-defined iteration methods (1), (4'), and (7') are combined with a search method that uses arguments

$$x + 2^k \cdot \Delta/(2i+1) \quad \text{for} \begin{cases} i=0, 1, \ldots, k \\ k=0, 1, \ldots \end{cases} \tag{10}$$

until an argument t is found for which either

$$|f(t)| < |f(x)| \quad \text{or} \quad f(t) \cdot f(x) \leq 0.$$

The value of Δ used internally is $\Delta = \min(0.1, |f(x)|)$.

2. If an interval (x_l, x_u) enclosing a root has been found, that is, $f(x_l) \cdot f(x_u) < 0$, then successive approximants from one of the iteration methods above must lie inside this interval. Otherwise, $(x_l + x_u)/2$ is used as the next guess. The interval bounds for this bisection method are updated in the course of calculation.

3. If no sign change has been located previously, the absolute argument change at a single iteration step is reduced to $\max(0.001, \Delta) \cdot \max(1, |x|)$ if necessary, in order to avoid overshooting and overflow problems.

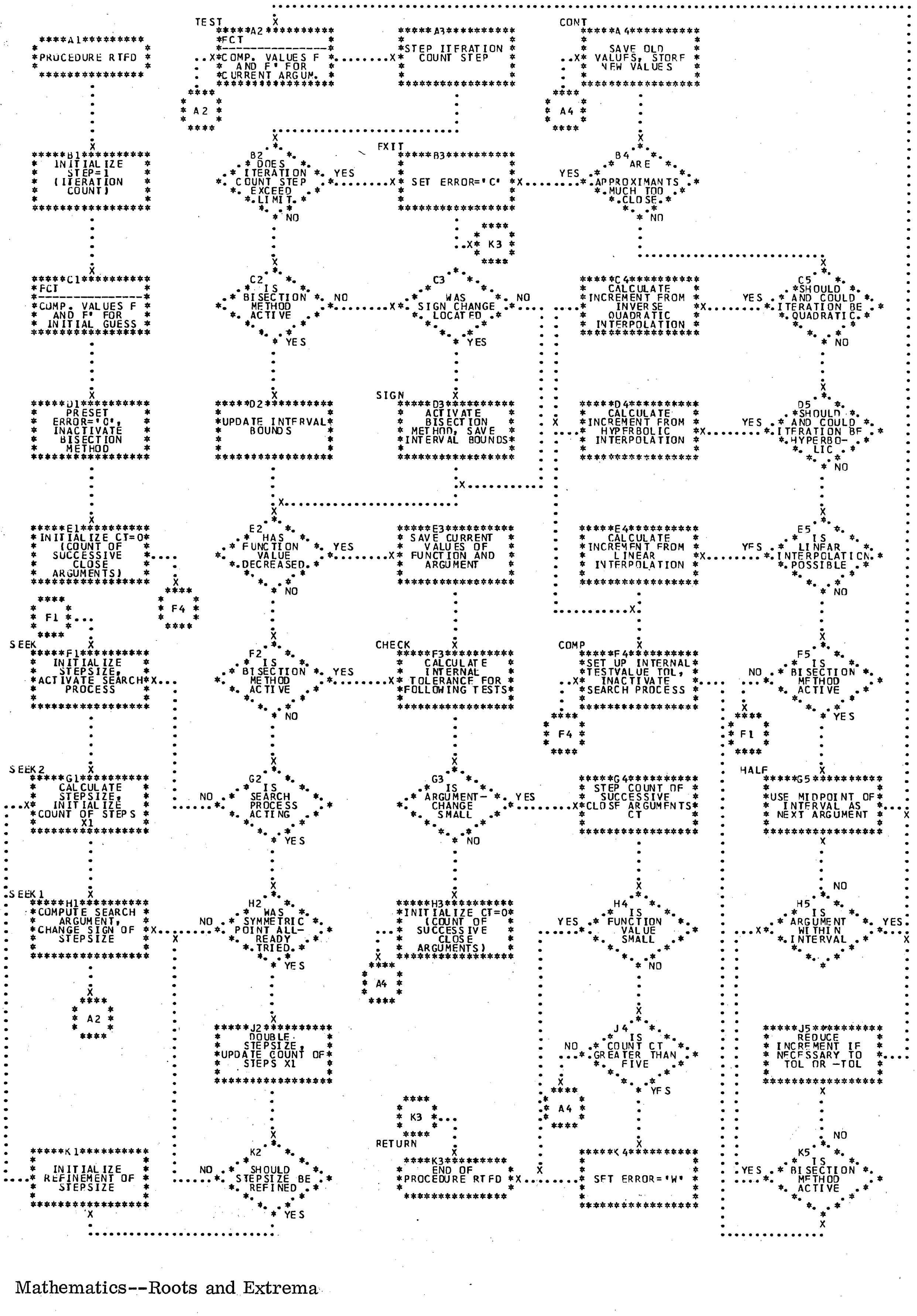
4. If, in case of no previous sign change, the iteration method fails to give an argument for which either $f(x_{i+1}) \cdot f(x_i) \leq 0$ or $|f(x_{i+1})| < |f(x_i)|$ the next approximation is calculated by search method (1).

5. Calculation of the first approximant is based on Newton's method in all cases, while for those following, the higher-order iteration methods are used if specified.

6. The convergence test used requires that both argument change and function value are absolutely less than or equal to $10^{-5} \max(1, |x|)$ in single precision and $10^{-12} \max(1, |x|)$ in double precison. If the argument change is absolutely less than or equal to this tolerance five times in sequence, while the function values are not small enough, then the currently best values x, f(x) and f'(x) are returned with ERROR='W'.

7. The iteration process is terminated with ERROR='C' if the number of function evaluations exceeds the user-specified limit LIMIT.

PROCEDURE RTFD REFINES AN INITIAL GUESS FOR A ROOT OF F(X)=0 USING LINEAR, INVERSE QUADRATIC OR HYPERBOLIC INTERPOLATION


A1 PROCEDURE RTFD
TEST A2 FCT COMP. VALUES F AND F' FOR CURRENT ARGUM.
A3 STEP ITERATION COUNT STEP
CONT A4 SAVE OLD VALUES, STORE NEW VALUES
B1 INITIALIZE STEP=1 (ITERATION COUNT)
B2 DOES ITERATION COUNT STEP EXCEED LIMIT.
YES
NO
EXIT B3 SET ERROR='C'
K3
B4 ARE APPROXIMANTS MUCH TOO CLOSE.
YES
NO
C1 FCT COMP. VALUES F AND F' FOR INITIAL GUESS
C2 IS BISECTION METHOD ACTIVE
NO
YES
C3 WAS SIGN CHANGE LOCATED
NO
YES
C4 CALCULATE INCREMENT FROM INVERSE QUADRATIC INTERPOLATION
C5 SHOULD AND COULD ITERATION BE QUADRATIC.
YES
NO
D1 PRESET ERROR='0', INACTIVATE BISECTION METHOD
D2 UPDATE INTERVAL BOUNDS
SIGN D3 ACTIVATE BISECTION METHOD, SAVE INTERVAL BOUNDS
D4 CALCULATE INCREMENT FROM HYPERBOLIC INTERPOLATION
D5 SHOULD AND COULD ITERATION BE HYPERBO-LIC
YES
NO
E1 INITIALIZE CT=0 (COUNT OF SUCCESSIVE CLOSE ARGUMENTS)
F1
F4
E2 HAS FUNCTION VALUE DECREASED.
YES
NO
E3 SAVE CURRENT VALUES OF FUNCTION AND ARGUMENT
E4 CALCULATE INCREMENT FROM LINEAR INTERPOLATION
E5 IS LINEAR INTERPOLATION POSSIBLE
YES
NO
SEEK F1 INITIALIZE STEPSIZE, ACTIVATE SEARCH PROCESS
F2 IS BISECTION METHOD ACTIVE
YES
NO
CHECK F3 CALCULATE INTERNAL TOLERANCE FOR FOLLOWING TESTS
COMP F4 SET UP INTERNAL TESTVALUE TOL, INACTIVATE SEARCH PROCESS
F4
F1
F5 IS BISECTION METHOD ACTIVE
NO
YES
SEEK2 G1 CALCULATE STEPSIZE, INITIALIZE COUNT OF STEPS X1
G2 IS SEARCH PROCESS ACTING
NO
YES
G3 IS ARGUMENT-CHANGE SMALL
YES
NO
G4 STEP COUNT OF SUCCESSIVE CLOSE ARGUMENTS CT
HALF G5 USE MIDPOINT OF INTERVAL AS NEXT ARGUMENT
SEEK1 H1 COMPUTE SEARCH ARGUMENT, CHANGE SIGN OF STEPSIZE
A2
H2 WAS SYMMETRIC POINT ALREADY TRIED.
NO
YES
H3 INITIALIZE CT=0 (COUNT OF SUCCESSIVE CLOSE ARGUMENTS)
A4
H4 IS FUNCTION VALUE SMALL
YES
NO
H5 IS ARGUMENT WITHIN INTERVAL
NO
YES
J2 DOUBLE STEPSIZE, UPDATE COUNT OF STEPS X1
J4 IS COUNT CT GREATER THAN FIVE
NO
YES
A4
J5 REDUCE INCREMENT IF NECESSARY TO TOL OR -TOL
K3
K1 INITIALIZE REFINEMENT OF STEPSIZE
K2 SHOULD STEPSIZE BE REFINED
NO
YES
RETURN K3 END OF PROCEDURE RTFD
K4 SET ERROR='W'
K5 IS BISECTION METHOD ACTIVE
YES
NO

• Subroutine DERE

```
DERE..
/**********************************************************************/
/*                                                                    */
/*      PERFORM ONE INTEGRATION STEP FOR A SYSTEM OF ORDINARY DIF-    */
/*      FERENTIAL EQUATIONS USING RATIONAL EXTRAPOLATION TECHNIQUE    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(F,N,H,X,Y,EPS),.
   DECLARE
        F ENTRY,                      /*Y' = F(X,Y) GIVEN ODE-SYSTEM */
        (ERROR EXTERNAL,CONV) CHARACTER(1),
        (EPS,YM(N),FMH,SQMH,FMM,SQMI,DSQMI)
        BINARY FLOAT,
        (H,X,Y(*),YI,DY(N),Z(N),DZ(N),LX,YC(N))
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (LH,HA,CI,BI,V,FO(N),FE(N),ZI,CMI,DI,U,
        DT(5*N))                      /*SINGLE PRECISION VERSION  /*S*/
 /*     DT(10*N))                     /*DOUBLE PRECISION VERSION  /*D*/
        BINARY FLOAT(53),
        (N,RR,CC,LN,DIAG,HSTEP,M,MM,I,J)
        BINARY FIXED,.
   LN   =N,.
   ERROR='S',.                        /*MARK ILLEGAL SPECIFICATION   */
   IF LN LE 0                         /*TEST SPECIFIED DIMENSION     */
   THEN GO TO EXIT,.
   LH   =H,.                          /*INIT. LOCAL STEPSIZE         */
   HSTEP=0,.                          /*INIT. COUNT HALVING STEPSIZE */
   IF LH= 0                           /*TEST SPECIFIED STEPSIZE      */
   THEN GO TO EXIT,.
   ERROR='0',.                        /*PRESET ERROR INDICATOR       */
   CALL F(X,Y,DY),.                   /*DERIVATIVE FOR INITIAL VALUES*/
   IF ERROR NE '0'
   THEN GO TO EXIT,.                  /*TERMINATE IF ERROR IN F(X,Y) */
                                      /*                             */
HALF..                                /*START OF ITERATION LOOP      */
   CONV ='H',.                        /*MARK FIRST APPROXIMATION     */
   DIAG =1,.                          /*INIT. DIAGONAL COUNT T-ARRAY */
   FMH  =0,.                          /*INIT. FLOATING EXTRAPOL.COUNT*/
                                      /*START OF EXTRAPOLATION LOOP  */
        DO M = 2 TO 16 BY 2,.         /*SINGLE PRECISION VERSION  /*S*/
 /*     DO M = 2 TO 28 BY 2,.         /*DOUBLE PRECISION VERSION  /*D*/
        FMH  =FMH+1,.                 /*UPDATE EXTRAPOLATION COUNT   */
        HA   =LH/FMH,.                /*CALCULATE INTERVAL SIZE      */
        FMM  =1,.
             DO MM = 1 TO M,.         /*COMP. DISCRETE APPROXIMATION */
                  DO I = 1 TO LN,.
                  YI   =Y(I),.
                  IF MM= 1            /*MODIFY MID-POINT RULE FOR    */
                  THEN DO,.           /*FIRST INTERVAL               */
                       IF CONV='H'    /*FOR THE VERY FIRST INTERVAL  */
                       THEN DO,.      /*INIT. VALUES FOR CONV. TEST  */
                            YC(I)=YI,.
                            YM(I)=ABS(YI),.
                            END,.
                       ZI,FE(I)=.50000000*DY(I),.
                       FO(I)=0,.      /*INIT. SUM OF DERIVATIVES     */
                       END,.
                  ELSE DO,.
                       ZI   =FO(I)+DZ(I),.
                       FO(I)=FE(I),.  /*UPDATE AND INTERCHANGE SUM OF*/
                       FE(I)=ZI,.     /*ODD/EVEN SPACED DERIVATIVES  */
                       END,.
                  Z(I),YI=HA*ZI+YI,.  /*COMP. APPROXIMATE FUNCTION   */
                  IF YM(I) LT ABS(YI) /*VALUE FOR LOCAL ARGUMENT LX  */
                  THEN YM(I)=ABS(YI),./*STORE MAX ABSOLUTE VALUE     */
                  END,.
             LX   =X+FMM*HA,.         /*COMP. LOCAL ARGUMENT         */
             FMM  =FMM+1,.
             CALL F(LX,Z,DZ),.        /*CALCULATE DERIVATIVE         */
             IF ERROR NE '0'
             THEN GO TO EXIT,.        /*TERMINATE IF ERROR IN F(X,Y) */
             END,.
        CONV ='C',.                   /*PRESET CONVERGENCE INDICATOR */
        SQMH =FMH*FMH,.               /*SQUARE EXTRAPOLATION COUNT   */
        HA   =HA*0.5,.
             DO I =1 TO LN,.          /*EXTRAPOLATION ON COMPONENTS  */
             V    =DT(I),.            /*SAVE OLD T-VALUE             */
             ZI,CI,DT(I)=Y(I)+HA*     /*STORE NEW T-VALUE            */
                  (.50000000*DZ(I)+FO(I)+FE(I)),.
             SQMI =SQMH,.             /*INIT. VARYING SQUARE NUMBER  */
             DSQMI=FMM,.              /*INIT. VARYING DECREMENT      */
             MM   =I,.
                  DO J = 2 TO DIAG,.
                  MM   =MM+LN,.
                  DSQMI=DSQMI-2,.     /*STEP ODD. INTEGER DECREMENT  */
                  SQMI =SQMI-DSQMI,.  /*COMPUTE NEXT LOWER SQUARE    */
                  BI   =SQMH*V,.
                  CMI  =CI*SQMI,.
                  DI   =BI-CMI,.      /*DENOMINATOR OF CENTRAL ALGOR.*/
                  U    =V,.
                  IF DI NE 0          /*TEST FOR ZERO DENOMINATOR    */
                  THEN DO,.           /*PERFORM RHOMBUS ALGORITHM    */
                       DI   =(CI-V)/DI,.
                       U    =CMI*DI,.
                       CI   =BI*DI,.
                       END,.
                  V    =DT(MM),.      /*SAVE OLD T-VALUE-DIFFERENCE  */
                  DT(MM)=U,.          /*STORE NEW T-VALUE-DIFFERENCE */
                  ZI   =ZI+U,.        /*COMP. NEW T-VALUE            */
                  END,.
             YI   =ABS(YC(I)-ZI),.
             IF YI LT ABS(U)          /*SET YI TO                    */
             THEN YI   =ABS(U),.      /*MAX(ABS(U),ABS(YC(I)-ZI))    */
             IF YI GT EPS*YM(I)       /*COMPONENTWISE CONVERGENCETEST*/
             THEN CONV ='1',.         /*NEGATE CONVERGENCE INDICATOR */
             YC(I)=ZI,.               /*STORE NEW COMPARISON VALUE   */
             END,.
        IF CONV='C'                   /*GLOBAL CONVERGENCE TEST      */
        THEN GO TO END,.
        ELSE IF DIAG LT 5             /*SINGLE PRECISION VERSION  /*S*/
 /*     ELSE IF DIAG LT 10            /*DOUBLE PRECISION VERSION  /*D*/
        THEN DIAG =DIAG+1,.           /*UPDATE DIAGONAL COUNT        */
```

```
        END,.
                                      /*END OF EXTRAPOLATION LOOP    */
   HSTEP=HSTEP+1,.                    /*UPDATE COUNT OF HALVING STEPS*/
   LH   =LH*0.5,.
   IF HSTEP LE 20                     /*MAXIMALLY 20 ITERATIONS WITH */
   THEN GO TO HALF,.                  /*REDUCED STEPSIZE             */
   ELSE GO TO EXIT,.                  /*TERMINATE IF NO CONVERGENCE  */
                                      /*END OF ITERATION LOOP        */
END..                                 /*SUCCESSFUL END OF OPERATION  */
   X    =X+LH,.                       /*RETURN ARGUMENT              */
   IF DIAG LE 4                       /*SINGLE PRECISION VERSION  /*S*/
 /*IF DIAG LE 7                       /*DOUBLE PRECISION VERSION  /*D*/
   THEN LH   =LH+LH,.                 /*DOUBLE STEPSIZE ESTIMATE     */
   H    =LH,.                         /*RETURN ADJUSTED STEPSIZE     */
        DO I = 1 TO LN,.
        Y(I) =YC(I),.                 /*RETURN EXTRAPOLATED FUNCTION-*/
        END,.                         /*VALUES                       */
EXIT..
   END,.                              /*END OF PROCEDURE DERE        */
```

Purpose:

DERE performs one integration step for a system of first order ordinary differential equations Y' = F(X, Y) with given initial values Y. The stepsize H is adjusted for accuracy requirements and speed considerations.

Usage:

CALL DERE (F, N, H, X, Y, EPS);

F - ENTRY
Given procedure for calculation of the derivatives.
This procedure must be supplied by the user.

Usage:
CALL F (T, Z, DZ);

T - BINARY FLOAT [(53)]
Given independent variable.

Z - BINARY FLOAT [(53)]
Given vector of dependent variables.

DZ - BINARY FLOAT [(53)]
Resultant vector of derivatives.

N - BINARY FIXED
Given dimension of the ODE system.

H - BINARY FLOAT [(53)]
Given suggested stepsize for current integration step.

X - BINARY FLOAT [(53)]
Given independent variable for initial values.
Resultant dependent variable for calculated values.

Y(N) - BINARY FLOAT [(53)]
Given initial values of vector Y for givenX.
Resultant calculated values of Y for resultant X.

EPS - BINARY FLOAT
Given relative tolerance for local error in calculated Y-values.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR = 'S' means $N \leq 0$ or $H = 0$
ERROR = '1' means no convergence was obtained with stepsizes $H/2^i$ for $i = 0, 1, \ldots, 20$

The last case may occur if stepsize $|H|$ is unrealistically large or if tolerance EPS is too small. Suggested values are $|H| = 1$ and $EPS \geq 10^{-5}$ in single precision and $EPS \geq 10^{-10}$ in double precision.

If ERROR is changed in the user-supplied procedure F(X, Y, DY) to a nonzero value, ERROR remains unchanged and DERE returns to the calling procedure immediately.

In all cases of a nonzero value of ERROR the parameters H, X, Y remain unchanged. The stepsize H of the integration step gets divided by a power of two if accuracy requirements are not met otherwise.

Method:

DERE uses a rational function for extrapolation and is based on the midpoint rule as the underlying discretization method.

For reference see:

R. Bulirsch and J. Stoer, "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods", Numerische Mathematik vol. 8, 1966, pp. 1-13.

Mathematical Background:

Notation

The problem is to solve the system of differential equations

$$y_1' = f_1\,(x, y_1, \ldots, y_n)$$
$$\vdots$$
$$y_n' = f_n\,(x, y_1, \ldots, y_n)$$

with given initial values

$$x_0,\ y_1(x_0) = y_{10}$$
$$\vdots$$
$$y_n(x_0) = y_{n0}$$

Using capital letters for vectors, this is written more compactly in vector form:

$$Y' = F(x, Y), \quad Y(x_0) = Y_0$$

Discretization method

The underlying discretization method proceeds as follows:

Set $h = H/2m$, $x_i = x_0 + ih$ and let $Z_i = Z(x_i, h)$

denote approximations to the exact value $Y(x_i)$ obtained with stepsize h by means of the midpoint rule:

$$Z_0 = Y_0, \quad Z_1 = Z_0 + hF(x_0, Z_0)$$

$$Z_{i+1} = Z_{i-1} + 2hF(x_i, Z_i) \quad \text{for } i = 1, 2, \ldots, 2m-1$$

Extrapolation is based on

$$T(h, x) = \frac{1}{2}(Z_{2m} + Z_{2m-1} + hF(x, Z_{2m}))$$

Under suitable differentiability assumptions the asymptotic expansion of T(h, x) proceeds with even powers of h:

$$T(h, x) = Y(x) + t_1(x)h^2 + t_2(x)h^4 + \ldots$$

Rational extrapolation method

Rational extrapolation is used to approximate

$$T(0, x) = Y(x)$$

Assume (h_k) to be a strictly decreasing sequence of stepsizes tending to zero and let

$$R_p^{(i)}(h) = \frac{p_0^{(i)} + p_1^{(i)}h^2 + \ldots + p_k^{(i)}h^{2k}}{q_0^{(i)} + q_1^{(i)}h^2 + \ldots + q_l^{(i)}h^{2l}},$$

$$k = \left[\frac{p}{2}\right], \quad l = p - k$$

be the rational function defined by p + 1 nodes:

$$R_p^{(i)}(h_j) = T(h_j, x), \quad j = i, i+1, \ldots, i+p$$

Then the extrapolated values $T_p^{(i)} = R_p^{(i)}(0)$ that approximate T(0, x) are obtained from the formulas

$$T_{-1}^{(i)} = 0$$

$$T_0^{(i)} = T(h_i, x)$$

$$T_k^{(i)} = T_{k-1}^{(i+1)} + \frac{T_{k-1}^{(i+1)} - T_{k-1}^{(i)}}{\left(\frac{h_i}{h_{i+k}}\right)^2 \left[1 - \frac{T_{k-1}^{(i+1)} - T_{k-1}^{(i)}}{T_{k-1}^{(i+1)} - T_{k-2}^{(i+1)}}\right] - 1}$$

for $k \geq 1$

The above formulas connect by a rhombus rule the elements

$T_{k-2}^{(i+1)}$, $T_{k-1}^{(i)}$, $T_{k-1}^{(i+1)}$, $T_k^{(i)}$ of the tableau

(T array):

$$\begin{array}{llll}
T_{-1}^{(0)} & & & \\
 & T_0^{(0)} & & \\
T_{-1}^{(1)} & & T_1^{(0)} & \\
\cdot & T_0^{(1)} & \cdot \; \cdots & T_p^{(0)} \\
\cdot & \cdot & \cdot & \\
\cdot & \cdot & \cdot & \\
\cdot & \cdot & T_1^{(p-1)} & \\
\cdot & \cdot & & \\
\cdot & T_0^{(p)} & & \\
T_{-1}^{(p+1)} & & &
\end{array}$$

Programming Considerations:

DERE uses the stepsize sequence

$$\left(h_1 = \frac{H}{2}, h_2 = \frac{H}{4}, \ldots, h_m = \frac{H}{2m}, \ldots\right)$$

for extrapolation.
The square numbers

$$\left(\frac{H/2}{h_{m-k}}\right)^2 = (m-k)^2 \text{ are}$$

generated successively using the identity

$$(l-1)^2 = l^2 - (2l-1)$$

which means that the next lower squares are obtained by subtracting decreasing odd integers.

To avoid repeated calculation of differences, the rhombus rule is modified to

$$D_{k-1}^{(m-k+1)} = \frac{C_{k-1}^{(m-k+1)} - T_{k-1}^{(m-k)}}{\left(\frac{H/2}{h_m}\right)^2 \Delta T_{k-1}^{(m-k)} - \left(\frac{H/2}{h_{m-k}}\right)^2 C_{k-1}^{(m-k+1)}}$$

$$\Delta T_k^{(m-k)} = \left(\frac{H/2}{h_{m-k}}\right)^2 C_{k-1}^{(m-k+1)} D_{k-1}^{(m-k+1)}$$

$$C_k^{(m-k)} = \left(\frac{H/2}{h_m}\right)^2 \Delta T_{k-1}^{(m-k)} D_{k-1}^{(m-k+1)}$$

for $k = 1, 2, \ldots, m$

Starting values are

$$\Delta T_0^{(i)} = C_0^{(i)} = T(h_i, x)$$

and the notation

$$\Delta T_k^{(i)} = T_k^{(i)} - T_{k-1}^{(i+1)}, \qquad C_k^{(i)} = T_k^{(i)} - T_{k-1}^{(i)}$$

implies

$$T_m^{(0)} = \sum_{k=0}^{m} \Delta T_k^{(m-k)}$$

The above formulas are evaluated successively for $m = 1, 2, \ldots$. Only one linear array is needed for storing the differences $\Delta T_k^{(m-k)}$.

Control of accuracy is done in a natural way: Comparing $T_{m-1}^{(0)}$ and $T_{m,}^{(0)}$ one increases the subscript m until this difference and the difference $\Delta T_m^{(0)}$ are small enough, which means less than the user-specified tolerance EPS times absolute maximum of the approximate function values Z_i obtained in the current interval of length H. This convergence test is applied componentwise.

The sensitivity of the extrapolation process to roundoff increases with the order of extrapolation. Therefore, the number of columns of the T array is limited to c = 5 for the single-precision version and to c = 10 for the double-precision version. The number of rows is limited to r = 8 and r = 14 respectively.

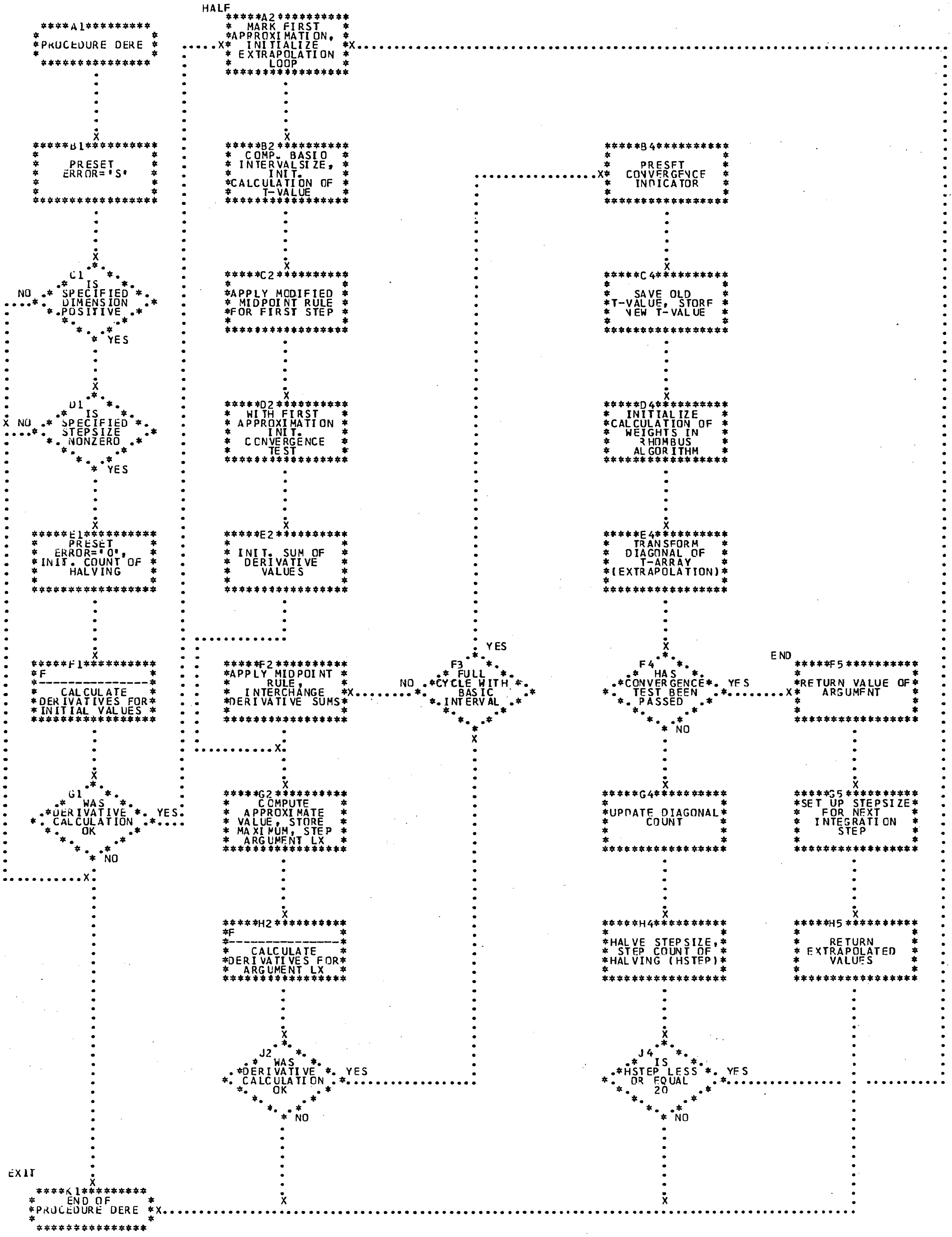
If k is not less than the maximum number of columns, the values $T_{c-1}^{(k-c+1)}$ are taken as successive approximations to the resulting values of Y. This continues up to $T_{c-1}^{(r-c+1)}$. If no convergence is reached at that point, the whole procedure is repeated with H/2 instead of H. DERE provides at most 20 iterations, each with half the stepsize of the one before. When there is no convergence, DERE returns to the calling procedure with ERROR='1' and parameters H, X, Y remain unchanged.

Adjustment of the stepsize H is a by-product of the above iteration process on length of stepsize. If convergence was attained with stepsize $H/2^j$, then $H/2^j$ is returned as the adjusted stepsize if at least 4 (7) extrapolation steps have been performed in single (double) precision to obtain the result values $Y(X+H/2^j)$ from input values Y(X), X.

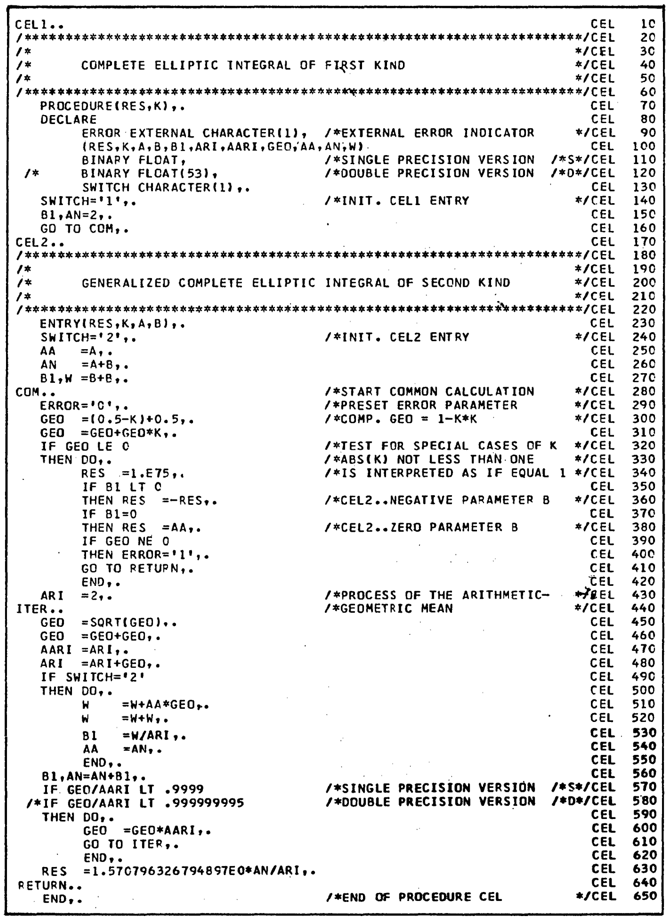
Otherwise, $H/2^{j-1}$ is returned as adjusted stepsize in order to speed up calculation time.

Since the extrapolation method does not necessarily work with a fixed order, adjustment of stepsize is uncritical. It does not critically affect accuracy, but only speed of computation.

PROCEDURE DERE PERFORMS ONE INTEGRATION STEP FOR A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS (INITIAL VALUE PROBLEM)


A1 PROCEDURE DERE
B1 PRESET ERROR='S'
C1 IS SPECIFIED DIMENSION POSITIVE
NO
YES
D1 IS SPECIFIED STEPSIZE NONZERO
NO
YES
E1 PRESET ERROR='0', INIT. COUNT OF HALVING
F1 F CALCULATE DERIVATIVES FOR INITIAL VALUES
G1 WAS DERIVATIVE CALCULATION OK
YES
NO
HALF
A2 MARK FIRST APPROXIMATION, INITIALIZE EXTRAPOLATION LOOP
B2 COMP. BASIC INTERVALSIZE, INIT. CALCULATION OF T-VALUE
C2 APPLY MODIFIED MIDPOINT RULE FOR FIRST STEP
D2 WITH FIRST APPROXIMATION INIT. CONVERGENCE TEST
E2 INIT. SUM OF DERIVATIVE VALUES
F2 APPLY MIDPOINT RULE, INTERCHANGE DERIVATIVE SUMS
G2 COMPUTE APPROXIMATE VALUE, STORE MAXIMUM, STEP ARGUMENT LX
H2 F CALCULATE DERIVATIVES FOR ARGUMENT LX
J2 WAS DERIVATIVE CALCULATION OK
YES
NO
F3 FULL CYCLE WITH BASIC INTERVAL
NO
YES
B4 PRESET CONVERGENCE INDICATOR
C4 SAVE OLD T-VALUE, STORE NEW T-VALUE
D4 INITIALIZE CALCULATION OF WEIGHTS IN RHOMBUS ALGORITHM
E4 TRANSFORM DIAGONAL OF T-ARRAY (EXTRAPOLATION)
F4 HAS CONVERGENCE TEST BEEN PASSED
YES
NO
G4 UPDATE DIAGONAL COUNT
H4 HALVE STEPSIZE, STEP COUNT OF HALVING (HSTEP)
J4 IS HSTEP LESS OR EQUAL 20
YES
NO
END
F5 RETURN VALUE OF ARGUMENT
G5 SET UP STEPSIZE FOR NEXT INTEGRATION STEP
H5 RETURN EXTRAPOLATED VALUES
EXIT
K1 END OF PROCEDURE DERE

- Subroutine CEL1/CEL2



```
CEL1..
/**********************************************************************/
/*                                                                    */
/*      COMPLETE ELLIPTIC INTEGRAL OF FIRST KIND                      */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(RES,K),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */
        (RES,K,A,B,B1,ARI,AARI,GEO,AA,AN,W)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION   /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION   /*D*/
        SWITCH CHARACTER(1),.
   SWITCH='1',.                       /*INIT. CEL1 ENTRY              */
   B1,AN=2,.
   GO TO COM,.
CEL2..
/**********************************************************************/
/*                                                                    */
/*      GENERALIZED COMPLETE ELLIPTIC INTEGRAL OF SECOND KIND         */
/*                                                                    */
/**********************************************************************/
   ENTRY(RES,K,A,B),.
   SWITCH='2',.                       /*INIT. CEL2 ENTRY              */
   AA   =A,.
   AN   =A+B,.
   B1,W =B+B,.
COM..                                 /*START COMMON CALCULATION      */
   ERROR='0',.                        /*PRESET ERROR PARAMETER        */
   GEO  =(0.5-K)+0.5,.                /*COMP. GEO = 1-K*K             */
   GEO  =GEO+GEO*K,.
   IF GEO LE 0                        /*TEST FOR SPECIAL CASES OF K   */
   THEN DO,.                          /*ABS(K) NOT LESS THAN ONE      */
        RES  =1.E75,.                 /*IS INTERPRETED AS IF EQUAL 1  */
        IF B1 LT 0
        THEN RES  =-RES,.             /*CEL2..NEGATIVE PARAMETER B    */
        IF B1=0
        THEN RES  =AA,.               /*CEL2..ZERO PARAMETER B        */
        IF GEO NE 0
        THEN ERROR='1',.
        GO TO RETURN,.
        END,.
   ARI  =2,.                          /*PROCESS OF THE ARITHMETIC-    */
ITER..                                /*GEOMETRIC MEAN                */
   GEO  =SQRT(GEO),.
   GEO  =GEO+GEO,.
   AARI =ARI,.
   ARI  =ARI+GEO,.
   IF SWITCH='2'
   THEN DO,.
        W    =W+AA*GEO,.
        W    =W+W,.
        B1   =W/ARI,.
        AA   =AN,.
        END,.
   B1,AN=AN+B1,.
   IF GEO/AARI LT .9999               /*SINGLE PRECISION VERSION   /*S*/
 /*IF GEO/AARI LT .999999995          /*DOUBLE PRECISION VERSION   /*D*/
   THEN DO,.
        GEO  =GEO*AARI,.
        GO TO ITER,.
        END,.
   RES  =1.57079632679489 7E0*AN/ARI,.
RETURN..
   END,.                              /*END OF PROCEDURE CEL          */
```

Purpose:

CEL1 computes the complete elliptic integral of the first kind:

$$\int_0^{\pi/2} dt/\sqrt{1-k^2\sin^2 t}, \quad 0 \le k < 1$$

Usage:

CALL CEL1 (RES, K);

RES - BINARY FLOAT [(53)]
Resultant value of elliptic integral.

K - BINARY FLOAT [(53)]
Given modulus of elliptic integral.

Purpose:

CEL2 computes the generalized elliptic integral of the second kind:

$$\int_0^{\pi/2} \frac{[a + (b - a)\sin^2 t]\,dt}{\sqrt{1 - k^2\sin^2 t}} \qquad 0 \le k < 1$$

Usage:

CALL CEL2 (RES, K, A, B);

RES - BINARY FLOAT [(53)]
Resultant value of elliptic integral.

K - BINARY FLOAT [(53)]
Given modulus of elliptic integral.

A - BINARY FLOAT [(53)]
Given primary term in numerator.

B - BINARY FLOAT [(53)]
Given secondary term in numerator.

Remarks:

If no errors are detected in the processing of data the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR = '1' means $|k| > 1$.

An input value of k with $|k| > 1$ is treated as if it were equal to 1. The value of k, however, remains unchanged.

Instead of ± infinity, the procedure returns $\pm 10^{75}$ as result values.

Method:

Calculation is based on the process of the arithmetic-geometric mean, combined with Landen's transformation.

For reference see:

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

M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, Applied Mathematics Series 55, National Bureau of Standards, 1964, pp. 597-599.

Mathematical Background:

Notation and equivalent definitions

Let k_c denote the complementary modulus defined through $k^2 + k_c^2 = 1$, $0 < k_c \leq 1$.

$$\mathrm{cel1}(k) = K(k) = \int_0^{\pi/2} \frac{dt}{\sqrt{1-k^2 \sin^2 t}}$$

$$= \int_0^{\infty} \frac{dx}{\sqrt{(1+x^2)(1+k_c^2 x^2)}}$$

$$\mathrm{cel2}(k;a,b) = \int_0^{\pi/2} \frac{a+(b-a)\sin^2 t}{\sqrt{1-k^2 \sin^2 t}}\, dt$$

$$= \int_0^{\infty} \frac{a+bx^2}{(1+x^2)\sqrt{(1+x^2)(1+k_c^2 x^2)}}\, dx$$

Important special cases of cel2 are the complete elliptic normal integrals:

$$K(k) = \mathrm{cel2}\ (k;1,1) = \int_0^{\pi/2} \frac{dt}{\sqrt{1-k^2 \sin^2 t}}$$

$$= \int_0^{1} \frac{dt}{\sqrt{(1-t^2)(1-k^2 t^2)}}$$

$$E(k) = \mathrm{cel2}\ (k;1,k_c^2) = \int_0^{\pi/2} \sqrt{1-k^2 \sin^2 t}\ dt$$

$$= \int_0^{1} \sqrt{\frac{1-k^2 t^2}{1-t^2}}\ dt$$

$$D(k) = \mathrm{cel2}(k;0,1) = \int_0^{\pi/2} \frac{\sin^2 t\ dt}{\sqrt{1-k^2 \sin^2 t}}$$

$$= \int_0^{1} \frac{t^2\ dt}{\sqrt{(1-t^2)(1-k^2 t^2)}}$$

$$B(k) = \mathrm{cel2}(k;1,0) = \int_0^{\pi/2} \frac{\cos^2 t}{\sqrt{1-k^2 \sin^2 t}}\ dt$$

$$= \int_0^{1} \sqrt{\frac{1-t^2}{1-k^2 t^2}}\ dt$$

Process of the arithmetic-geometric mean

Starting with the pair of numbers:

$$a = 2,\ g = 2k_c$$

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

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

This iteration process is stopped at the N^{th} step when $a_N = g_N$ to the degree of accuracy of the finite arithmetic employed.

In case cel2 the sequences (A_i), (B_i) are also needed. They are defined by means of

$$A_0 = A, \quad B_0 = 2B$$

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

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

Result values obtained are

$$\mathrm{cel1}(k) = \frac{\pi}{2} \cdot \frac{2^{N+1}}{a_N}$$

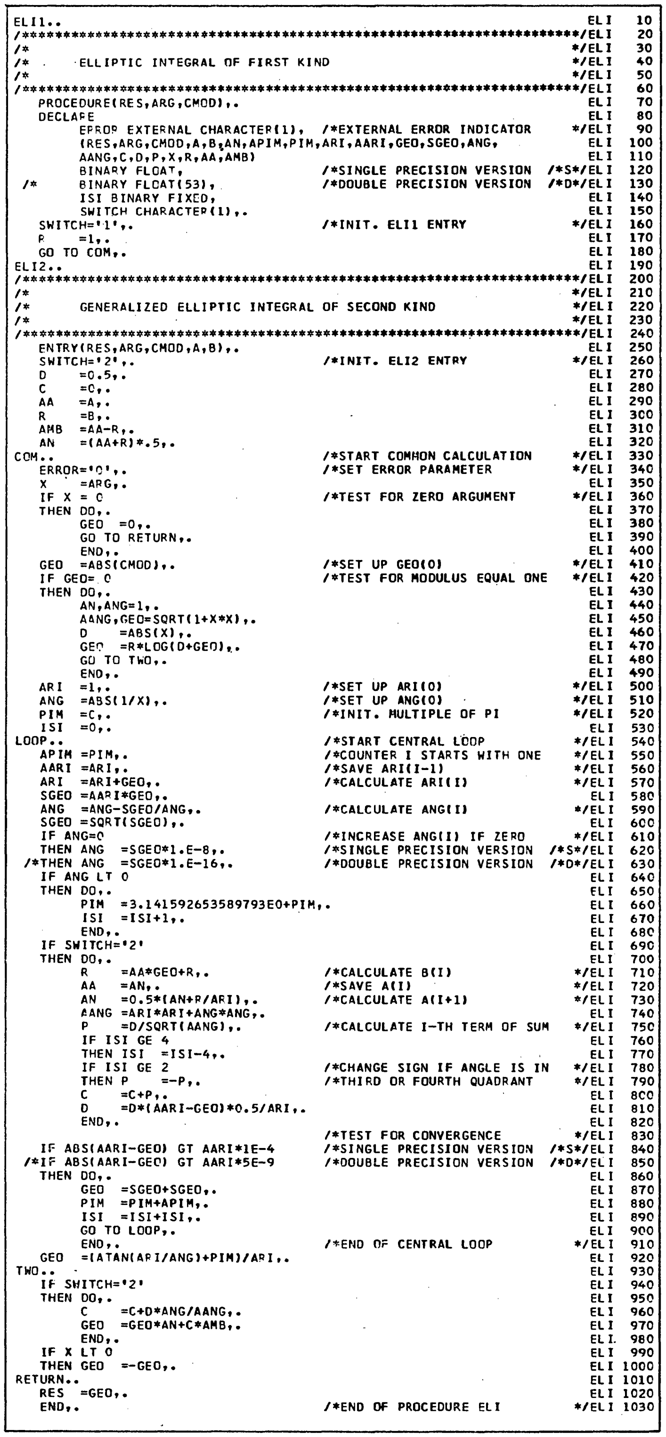
$$\mathrm{cel2}(k, A, B) = \frac{\pi}{2} \cdot \frac{A_{N+1}}{a_N}$$

Programming Considerations:

The equality $a_N = g_N$ must be interpreted as $|a_N - g_N|$ is less than $a_N \cdot 10^{-D}$, where D is the number of decimal digits in the mantissa of floating-point numbers.

Since the sequences $(2^{-n} \cdot a_n)$, $(2^{-n} \cdot g_n)$ converge quadratically to the same limit (arithmetic-geometric mean), the above test 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 geometric mean.

Subroutine ELI1/ELI2



```
ELI1..
/**********************************************************************/
/*                                                                    */
/*  .   ELLIPTIC INTEGRAL OF FIRST KIND                               */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(RES,ARG,CMOD),.
   DECLAFE
        EPROP EXTERNAL CHARACTEP(1),  /*EXTERNAL ERROR INDICATOR      */
        (RES,ARG,CMOD,A,B,AN,APIM,PIM,ARI,AARI,GEO,SGEO,ANG,
        AANG,C,D,P,X,R,AA,AMB)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        ISI BINARY FIXED,
        SWITCH CHARACTEP(1),.
   SWITCH='1',.                       /*INIT. ELI1 ENTRY              */
   P    =1,.
   GO TO COM,.
ELI2..
/**********************************************************************/
/*                                                                    */
/*      GENERALIZED ELLIPTIC INTEGRAL OF SECOND KIND                  */
/*                                                                    */
/**********************************************************************/
   ENTRY(RES,ARG,CMOD,A,B),.
   SWITCH='2',.                       /*INIT. ELI2 ENTRY              */
   D    =0.5,.
   C    =0,.
   AA   =A,.
   R    =B,.
   AMB  =AA-R,.
   AN   =(AA+R)*.5,.
COM..                                 /*START COMMON CALCULATION      */
   ERROR='0',.                        /*SET ERROR PARAMETER           */
   X    =ARG,.
   IF X = 0                           /*TEST FOR ZERO ARGUMENT        */
   THEN DO,.
        GEO  =0,.
        GO TO RETURN,.
        END,.
   GEO  =ABS(CMOD),.                  /*SET UP GEO(0)                 */
   IF GEO= 0                          /*TEST FOR MODULUS EQUAL ONE    */
   THEN DO,.
        AN,ANG=1,.
        AANG,GEO=SQRT(1+X*X),.
        D    =ABS(X),.
        GEO  =R*LOG(D+GEO),.
        GO TO TWO,.
        END,.
   ARI  =1,.                          /*SET UP ARI(0)                 */
   ANG  =ABS(1/X),.                   /*SET UP ANG(0)                 */
   PIM  =C,.                          /*INIT. MULTIPLE OF PI          */
   ISI  =0,.
LOOP..                                /*START CENTRAL LOOP            */
   APIM =PIM,.                        /*COUNTER I STARTS WITH ONE     */
   AARI =ARI,.                        /*SAVE ARI(I-1)                 */
   ARI  =ARI+GEO,.                    /*CALCULATE ARI(I)              */
   SGEO =AARI*GEO,.
   ANG  =ANG-SGEO/ANG,.               /*CALCULATE ANG(I)              */
   SGEO =SQRT(SGEO),.
   IF ANG=0                           /*INCREASE ANG(I) IF ZERO       */
   THEN ANG  =SGEO*1.E-8,.            /*SINGLE PRECISION VERSION  /*S*/
 /*THEN ANG  =SGEO*1.E-16,.           /*DOUBLE PRECISION VERSION  /*D*/
   IF ANG LT 0
   THEN DO,.
        PIM  =3.141592653589793E0+PIM,.
        ISI  =ISI+1,.
        END,.
   IF SWITCH='2'
   THEN DO,.
        R    =AA*GEO+R,.              /*CALCULATE B(I)                */
        AA   =AN,.                    /*SAVE A(I)                     */
        AN   =0.5*(AN+R/ARI),.        /*CALCULATE A(I+1)              */
        AANG =ARI*ARI+ANG*ANG,.
        P    =D/SQRT(AANG),.          /*CALCULATE I-TH TERM OF SUM    */
        IF ISI GE 4
        THEN ISI  =ISI-4,.
        IF ISI GE 2                   /*CHANGE SIGN IF ANGLE IS IN    */
        THEN P    =-P,.               /*THIRD OR FOURTH QUADRANT      */
        C    =C+P,.
        D    =D*(AARI-GEO)*0.5/ARI,.
        END,.
                                      /*TEST FOR CONVERGENCE          */
   IF ABS(AARI-GEO) GT AARI*1E-4      /*SINGLE PRECISION VERSION  /*S*/
 /*IF ABS(AARI-GEO) GT AARI*5E-9      /*DOUBLE PRECISION VERSION  /*D*/
   THEN DO,.
        GEO  =SGEO+SGEO,.
        PIM  =PIM+APIM,.
        ISI  =ISI+ISI,.
        GO TO LOOP,.
        END,.                         /*END OF CENTRAL LOOP           */
   GEO  =(ATAN(ARI/ANG)+PIM)/ARI,.
TWO..
   IF SWITCH='2'
   THEN DO,.
        C    =C+D*ANG/AANG,.
        GEO  =GEO*AN+C*AMB,.
        END,.
   IF X LT 0
   THEN GEO  =-GEO,.
RETURN..
   RES  =GEO,.
   END,.                              /*END OF PROCEDURE ELI          */
```

Purpose:

ELI1 computes the incomplete elliptic integral of first kind for given values of an argument x and complementary modulus ck.

$$\text{eli1}\ (x,\ ck) = \int_0^x \frac{dt}{\sqrt{(1+t^2)(1+ck^2 \cdot t^2)}}$$

Usage:

CALL ELI1 (RES, ARG, CMOD);

RES - BINARY FLOAT [(53)]
Resultant value of elliptic integral.

ARG - BINARY FLOAT [(53)]
Given argument of elliptic integral.

CMOD - BINARY FLOAT [(53)]
Given complementary modulus of elliptic integral.

Purpose:

ELI2 computes the generalized incomplete elliptic integral of second kind for given values of an argument x, complementary modulus ck, and constants a and b.

$$\text{eli2}\ (x,\ ck;\ a,\ b) = \int_0^x \frac{(a+bt^2)\ \ dt}{(1+t^2)\sqrt{(1+t^2)\,(1+ck^2 \cdot t^2)}}$$

Usage:

CALL ELI2 (RES, ARG, CMOD, A, B);

RES - BINARY FLOAT [(53)]
Resultant value of elliptic integral.

ARG - BINARY FLOAT [(53)]
Given argument of elliptic integral.

CMOD - BINARY FLOAT [(53)]
Given complementary modulus of elliptic integral.

A - BINARY FLOAT [(53)]
Given primary term in numerator (see "Purpose").

B - BINARY FLOAT [(53)]
Given secondary term in numerator (see "Purpose").

Remarks:

Modulus k and complementary modulus ck satisfy the relation $k^2 + ck^2 = 1$. Internally, ck is needed for calculation rather than k. Therefore, ck is used as input parameter. This allows the modulus k to be any pure imaginary or real number such that $k^2 \leq 1$.

Method:

Calculation is based on the process of the arithmetic-geometric mean, combined with descending Landen's transformation.

For reference see:

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

Mathematical Background:

Notation and equivalent definitions:

$$\mathrm{eli1}(x,ck) = \int_0^x \frac{dt}{\sqrt{(1+t^2)(1+ck^2\cdot t^2)}}$$

$$= \int_0^{\arctan x} \frac{dt}{\cos t\sqrt{1+ck^2\cdot \tan^2 t}} = \int_0^{\arctan x} \frac{dt}{\sqrt{1-k^2 \sin^2 t}}$$

$$\mathrm{eli2}(x,\ ck,\ a,\ b) = \int_0^x \frac{(a+bt^2)\,dt}{(1+t^2)\sqrt{(1+t^2)(1+ck^2\cdot t^2)}}$$

$$= \int_0^{\arctan x} \frac{(a+b\tan^2 t)\,dt}{\sqrt{(1+\tan^2 t)(1+ck^2\cdot\tan^2 t)}}$$

$$= \int_0^{\arctan x} \frac{(a+(b-a)\sin^2 t)\,dt}{\sqrt{1-k^2\sin^2 t}}$$

Important special cases are:

$$F(\varphi,k) = \mathrm{eli1}\ (\tan\varphi,\ ck) = \int_0^{\varphi} \frac{dt}{\sqrt{1-k^2\sin^2 t}}$$

$$= \mathrm{eli2}\ (\tan\varphi,\ ck;\ 1,\ 1)$$

$$E(\varphi,k) = \mathrm{eli2}(\tan\varphi,\ ck;\ 1,\ ck^2) = \int_0^{\varphi} \sqrt{1-k^2\sin^2 t}\ \ dt$$

$$D(\varphi,k) = \frac{F(\varphi,k) - E(\varphi,k)}{k^2} = \mathrm{eli2}(\tan\varphi, ck;\ 0,1)$$

$$= \int_0^{\varphi} \frac{\sin^2 t\,dt}{\sqrt{1-k^2\sin^2 t}}$$

$$B(\varphi,k) = \frac{E(\varphi,k) - ck^2 F(\varphi,k)}{k^2} = \mathrm{eli2}(\tan\varphi, ck;1,0)$$

$$= \int_0^{\varphi} \frac{\cos^2 t\,dt}{\sqrt{1-k^2\sin^2 t}}$$

Process of the arithmetic-geometric mean

Starting with $ari_0=1$, $geo_0= |ck|$, the sequences (ari_n), (geo_n) are generated using the recursion formulas

$$ari_{n+1} = ari_n + geo_n \qquad (1)$$

$$geo_{n+1} = 2\sqrt{ari_n \cdot geo_n} \qquad (2)$$

This iterative process is stopped at the N^{th} step, when $ari_N = geo_N$ to the degree of accuracy of the finite arithmetic employed.

Descending Landen's transformation

For the descending Landen transformation the modular angle α defined by $k = \sin\alpha$ decreases, while the amplitudinal angle φ defined by $x = \tan\varphi$ increases.

Successive values of α and φ are combined as follows:

$$(1+\sin\alpha_1)(1+\cos\alpha) = 2 \qquad \alpha_1 < \alpha \qquad (3)$$

$$\tan(\varphi_1 - \varphi) = \cos\alpha\cdot\tan\varphi \qquad \varphi_1 > \varphi \qquad (4)$$

Expressed in terms of argument x and complementary modulus ck, these equations read

$$ck_1 = \frac{2\sqrt{ck}}{1+ck} \qquad (5)$$

$$x_1 = \frac{(1+ck)x}{1-ck\cdot x^2} \qquad (6)$$

For values of argument and modulus that are connected by (5) and (6) we have

$$\text{eli1}\,(x, ck) = \frac{1}{1+ck}\,\text{eli1}\,(x_1, ck_1) \tag{7}$$

$$\text{eli2}\,(x, ck; a, b) = \frac{1}{1+ck}\,\text{eli2}(x_1, ck_1; a_1, b_1) + \frac{(a-b)}{2} \cdot \frac{x_1}{\sqrt{1+x_1^2}}\;) \tag{8}$$

where

$$a_1 = (a+b)/2 \tag{9}$$

$$b_1 = \frac{1}{1+ck}\,(b+a \cdot ck) \tag{10}$$

The sign determination of $\frac{x_1}{\sqrt{1+x_1 2}} = \sin\varphi_1$

must be done such that $\varphi_1 = \arctan x_1$ is monotonically increasing ($\varphi_1 > \varphi$).

Final iteration process

We set: $x_0 = |x|$ and $ang_0 = 1/x_0$

$$x_i = \frac{ari_i}{ang_i} \tag{11}$$

$$ck_i = \frac{geo_i}{ari_i} \tag{12}$$

Furthermore, in case eli2 we use:

$$A_i = a_i, \; B_i = b_i \cdot ari_i$$

then:

$$A_0 = a, \; B_0 = b$$

$$A_{i+1} = 1/2\,(A_i + \frac{B_i}{ari_i}) \tag{13}$$

$$B_{i+1} = B_i + geo_i \cdot A_i \tag{14}$$

Successive application of the descending Landen transformation gives

$$\text{eli1}(x, ck) = \frac{ari_0}{ari_1}\,\text{eli1}(x_1, ck_1)$$

$$= \frac{ari_0}{ari_1} \cdot \frac{ari_1}{ari_2}\,\text{eli1}(x_2, ck_2) \;.\;.\;.$$

$$= \frac{ari_0}{ari_N}\;\text{eli1}(x_N, ck_N)$$

$$\text{eli2}(x, ck; a, b) = \frac{ari_0}{ari_1}\,\text{eli2}(x_1, ck_1; a_1, b_1)$$

$$+ \frac{a-b}{2} \cdot \frac{\sin\varphi_1}{ari_1}$$

$$= \frac{ari_0}{ari_2}\,\text{eli2}\,(x_2, ck_2; a_2, b_2)$$

$$+ \frac{a-b}{2}\left(\frac{\sin\varphi_1}{ari_1} + \frac{ari_0 - geo_0}{ari_1}\,\frac{\sin\varphi_2}{2 \cdot ari_2}\right)$$

$$= \cdot\;\cdot\;\cdot$$

$$= \frac{ari_0}{ari_N}\;\text{eli2}\,(x_N, ck_N; a_N, b_N) + \text{SUM}$$

where:

$$\text{SUM} = \frac{a-b}{2}\left(\frac{1}{ari_1}\,\sin\varphi_1 + \frac{1}{ari_2} \cdot \frac{ari_0 - geo_0}{ari_1} \cdot \frac{\sin\varphi_2}{2} + \;.\;.\;. + \frac{1}{ari_N} \cdot \frac{ari_0 - geo_0}{ari_1} \;.\;.\;. \frac{ari_{N-2} - geo_{N-2}}{ari_{N-1}} \cdot \frac{\sin\varphi_N}{2^{N-1}}\right)$$

Since $ck_N = 1$ to working accuracy:

$$\text{eli1}\,(x_N, ck_N) = \varphi_N, \text{ where } \tan\varphi_N = \frac{ari_N}{ang_N}$$

$$\text{eli2}\,(x_N,\ ck_N;\ a_N,\ b_N) = \frac{a_N+b_N}{2} \cdot \varphi_N$$

$$+ \frac{a_N-b_N}{2} \sin \varphi_N \cdot \cos \varphi_N$$

The final result is

$$\text{eli1}\,(x,\ ck) = \frac{\varphi_N}{\text{ari}_N}$$

$$\text{eli2}\,(x,\ ck; a, b) = \frac{A_{N+1}}{\text{ari}_N}\,\varphi_N + \text{SUM}$$

$$+ \frac{1}{\text{ari}_N}\left(\frac{a_N-b_N}{2}\right) \sin \varphi_N \cdot \cos \varphi_N$$

Degenerate cases of argument and modulus

$x = 0$ gives result eli2 = 0

$$ck = 0 \text{ gives result eli2} = \left(b \cdot \ln\left(|x| + \sqrt{1+x^2}\right) + (a-b)\right) \frac{x \,\text{sgn}\, x}{\sqrt{1+x^2}}$$

Programming Considerations:

The equality $\text{ari}_N = \text{geo}_N$ must be interpreted as ' $\text{ari}_N - \text{geo}_N|$ is less than $\text{ari}_N \cdot 10^{-D}$, where D is the number of decimal digits in the mantissa of floating-point numbers.

Since the sequences $(\text{ari}_n \cdot 2^{-n})$, $(\text{geo}_n \cdot 2^{-n})$ converge quadratically to the same limit (arithmetic-geometric mean), the above test may be replaced by comparing

$|\text{ari}_{N-1} - \text{geo}_{N-1}|$ against $\text{ari}_{N-1} \cdot 10^{-D/2}$, thus saving one calculation of the geometric mean.

- Subroutine JELF

```
JELF..
/**********************************************************************/
/*                                                                    */
/*      JACOBIAN ELLIPTIC FUNCTIONS SN, CN, DN                        */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(SN,CN,DN,X,SCK),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */
        (SN,CN,DN,X,SCK,CM,Y,LSN,LCN,LDN,K,ARI(12),GEO(12),A,B,C,D)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION  /*D*/
        (I,J) BINARY FIXED,.
   ERROR='0',.
   CM   =SCK,.
   Y    =X,.
   IF CM= 0                           /*TEST VALUE OF MODULUS         /
   THEN DO,.                          /*DEGENERATE CASE SCK = C       */
        LCN,LDN=1/COSH(Y),.
        LSN  =TANH(Y),.
        GO TO RETURN,.
        END,.
   IF CM LT 0
   THEN DO,.                          /*MODULUS TRANSFORMATION        */
        K    =(0.5-CM)+0.5,.
        CM   =-CM/K,.
        K    =SQRT(K),.
        Y    =K*Y,.
        END,.
   C,LDN=1,.                          /*PROCESS OF THE ARITHMETIC-    */
        DO I=1 TO 12,.                /*GEOMETRIC MEAN                */
        ARI(I),LCN=C,.
        GEO(I),CM=SQRT(CM),.
        C    =.5*(LCN+CM),.
        IF ABS(LCN-CM) LE 1E-4*LCN    /*SINGLE PRECISION VERSION  /*S*/
 /*     IF ABS(LCN-CM) LE 5E-9*LCN    /*DOUBLE PRECISION VERSION  /*D*/
        THEN GO TO CONV,.
        CM   =CM*LCN,.
        END,.
CONV..                                /*INIT. INVERSE GAUSS-          */
   Y    =Y*C,.                        /*TRANSFORMATION                */
   LSN,D=SIN(Y),.
   LCN  =COS(Y),.
   IF LSN= 0
   THEN GO TO TEST,.
   A    =LCN/LSN,.
   C    =A*C,.
        DO J =I TO 1 BY -1,.          /*INVERSE GAUSS-TRANSFORMATION */
        B    =ARI(J),.
        A    =A*C,.
        C    =LDN*C,.
        LDN  =(GEO(J)+A)/(B+A),.
        A    =C/B,.
        END,.
   LSN  =SQRT(1/(1+C*C)),.
   IF D LT 0
   THEN LSN  =-LSN,.
   LCN  =C*LSN,.
TEST..                                /*INVERSE MODULUS-TRANSFORMAT. */
   IF SCK LT 0
   THEN DO,.
        A    =LDN,.
        LDN  =LCN,.
        LCN  =A,.
        LSN  =LSN/K,.
        END,.
RETURN..                              /*RETURN RESULT VALUES          */
   SN   =LSN,.
   CN   =LCN,.
   DN   =LDN,.
   END,.                              /*END OF PROCEDURE JELF         */
```

Purpose:

JELF calculates the three Jacobian elliptic functions SN, CN, DN.

Usage:

CALL JELF (SN, CN, DN, X, SCK);

SN - BINARY FLOAT [(53)]
Resultant value of the sine of the amplitude.

CN - BINARY FLOAT [(53)]
Resultant value of the cosine of the amplitude.

DN - BINARY FLOAT [(53)]
Resultant value of the delta of the amplitude.

X - BINARY FLOAT [(53)]
Given argument of Jacobian elliptic functions.

SCK - BINARY FLOAT [(53)]
Given square of complementary modulus.

Remarks:

The values of SN, CN, DN are frequently needed together. Therefore, procedure JELF computes all three of them. This is no disadvantage, since computation of all three result values is no more complicated than computation of any one of them. The value SCK is chosen as an input parameter in order to allow for complex values of ck (k is not restricted to $k^2 \le 1$).

Method:

The calculation is based on the process of the arithmetic-geometric mean together with Gauss' transformation.

For reference see:

R. Bulirsch, "Numerical Calculation of Elliptic Integrals and Elliptic Functions", Numerische Mathematik, vol. 7, 1965, pp. 78-90.

Mathematical Background:

Notation and definition

The value k is the modulus, ck is the complementary modulus, and sck is the square of the complementary modulus.

$$sck = ck^2 = 1-k^2 \qquad -\infty < sck < \infty$$

The three Jacobian elliptic functions arise as inverse functions of elliptic integrals.
Set:

$$x = F(\varphi,k) = \int_0^{\varphi} \frac{dt}{\sqrt{1-k^2 \sin^2 t}}$$

Then φ is called the amplitude of x.

$$\varphi = \text{am}\ (x,k) \qquad (1)$$

Jacobi's functions are defined through

$$sn(x,k) = \sin\varphi = \sin \text{am}\ (x,k) \qquad (2)$$

$$cn(x,k) = \cos\varphi = \cos \text{am}\ (x,k) \qquad (3)$$

$$dn(x,k) = \sqrt{1-k^2 \sin^2\varphi} \qquad (4)$$

The degenerate case sck = 0 (that is, $|k| = 1$) must be treated separately:

$$sn(x,1) = \tanh x;$$

$$cn(x,1) = dn\ (x,1) = 1/\cosh x$$

Jacobi's modulus transformation, applied to negative values of sck, gives

$$sn(x,k) = 1/k \cdot sn\ (kx, 1/k) \qquad (5)$$

$$cn(x,k) = dn(kx, 1/k) \qquad (6)$$

$$dn(x,k) = cn(kx, 1/k) \qquad (7)$$

Process of the arithmetic-geometric mean

Starting with $ari_1 = 1$, $geo_1 = \sqrt{sck}$, the sequences (ari_n), (geo_n) are generated using the recursion formulas

$$ari_{n+1} = (ari_n + geo_n)/2 \qquad (8)$$

$$geo_{n+1} = \sqrt{ari_n \cdot geo_n} \qquad (9)$$

Numerical experience shows that eleven iterations are sufficient to obtain convergence, to full working accuracy, for all values of the squared complementary modulus that may be represented in floating point. The iteration process is stopped at the N^{th} step, as soon as $ari_{N+1} - geo_{N+1}$ is negligibly small.

Gauss transformation

Gauss' transformation gives

$$F(\varphi_1, k_1) = (1+k)\ F(\varphi,k) \qquad (10)$$

for values of modulus and amplitudinal angle that are combined through

$$k_1 = \frac{2\sqrt{k}}{1+K} \qquad (11)$$

and

$$\sin\varphi_1 = \frac{(1+k)\sin\varphi}{1+k\sin^2\varphi} \qquad (12)$$

Inversion of this transformation results in

$$F(\varphi,k) = (1+k_1)\ F(\varphi_1, k_1) \qquad (10')$$

where:

$$\sin\varphi = \frac{(1+k_1)\sin\varphi_1}{1+k_1\sin^2\varphi_1} \qquad (11')$$

and

$$k = \frac{2\sqrt{k_1}}{1+k_1} \qquad (12')$$

Inversion of $F(\varphi, k)$

Successive application of transformation (10'), with

$$ck_i = \frac{geo_{i+1}}{ari_{i+1}} \qquad (13)$$

leads to $F(\varphi,k) = (1+k_1)\ldots(1+k_N)\,F(\varphi_N, k_N)$.

Equation (12') implies that $k_{i+1} = \frac{1-ck_i}{1+ck_i}$

and that

$$1 + k_{i+1} = \frac{ari_{i+1}}{ari_{i+2}}$$

If $k_N = 0$, it follows that

$$x = F(\varphi,k) = \frac{ari_1}{ari_{N+1}}$$

$$F(\varphi_N, k_N) = \frac{ari_1}{ari_{N+1}}\varphi_N \qquad (14)$$

or $\varphi_N = ari_{N+1}\cdot x$

Back transformation of φ_N

To obtain the Jacobian elliptic functions, the inverse transformation must be performed on φ_N. Equation (11') implies

$$\cot\varphi = \frac{1}{1+k_1}\cot\varphi_1\sqrt{1-k_1^2\sin^2\varphi_1}$$

or generally

$$ari_n\cot\varphi_{n-1} = ari_{n+1}\cot\varphi_n\sqrt{1-k_n^2\sin^2\varphi_n} \qquad (15)$$

From (11') and (12') it follows that

$$\sqrt{1-k_n^2\sin^2\varphi_n} = \frac{1-k_{n+1}\sin^2\varphi_{n+1}}{1+k_{n+1}\sin^2\varphi_{n+1}}$$

$$= \frac{\cot^2\varphi_{n+1} + 1-k_{n+1}}{\cot^2\varphi_{n+1} + 1+k_{n+1}} \qquad (16)$$

$$= \frac{geo_{n+1} + ari_{n+2}\cdot\cot^2\varphi_{n+1}}{ari_{n+1} + ari_{n+2}\cdot\cot^2\varphi_{n+1}}$$

since

$$\frac{1-k_{n+1}}{1+k_{n+1}} = ck_n = \frac{geo_{n+1}}{ari_{n+1}}$$

and

$$1+k_{n+1} = \frac{ari_{n+1}}{ari_{n+2}}$$

Final iteration scheme

Setting $c_{N+1} = ari_{N+1}\cdot\cot\varphi_N$, with $d_{N+1} = 1$, the following iteration is performed for n = N, N-1, . . . , 1:

$$c_n = d_{n+1}\cdot c_{n+1}$$

$$d_n = \frac{c_{n+1}^2/ari_{n+2} + geo_{n+1}}{c_{n+1}^2/ari_{n+2} + ari_{n+1}}$$

The final result is

$$c_1 = \cot\varphi$$

$$d_1 = \sqrt{1-k^2\sin^2\varphi}$$

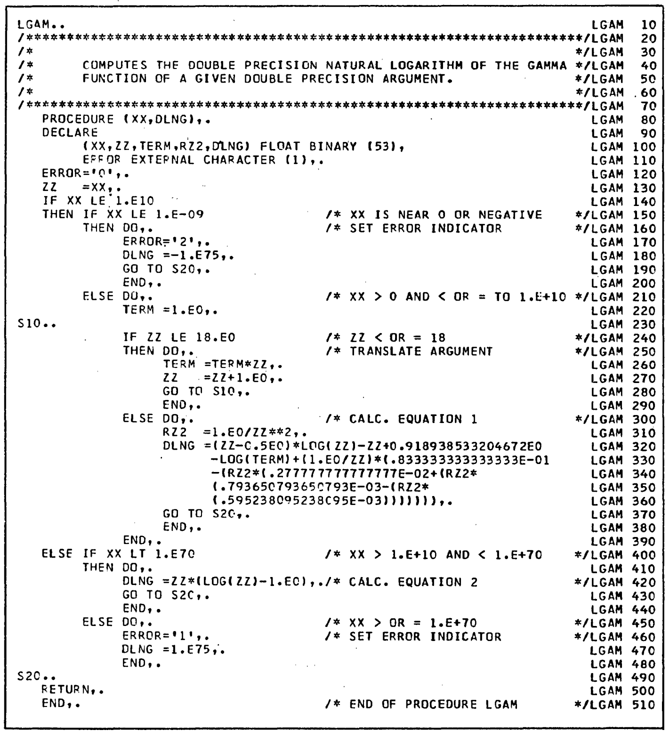
and therefore:

$$sn(x,k) = \frac{1}{\sqrt{1+c_1^2}} = \sin\varphi$$

$$cn(x,k) = sn\cdot c_1 = \cos\varphi$$

$$dn(x,k) = d_1 = \sqrt{1-k^2\sin^2\varphi}$$

- Subroutine LGAM



```
LGAM..
/*********************************************************************/
/*                                                                   */
/*      COMPUTES THE DOUBLE PRECISION NATURAL LOGARITHM OF THE GAMMA */
/*      FUNCTION OF A GIVEN DOUBLE PRECISION ARGUMENT.               */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (XX,DLNG),.
   DECLARE
        (XX,ZZ,TERM,RZ2,DLNG) FLOAT BINARY (53),
        EFFOR EXTEPNAL CHARACTER (1),.
   ERROR='0',.
   ZZ   =XX,.
   IF XX LE 1.E10
   THEN IF XX LE 1.E-09               /* XX IS NEAR 0 OR NEGATIVE    */
        THEN DO,.                     /* SET ERROR INDICATOR         */
             ERROR='2',.
             DLNG =-1.E75,.
             GO TO S20,.
             END,.
        ELSE DO,.                     /* XX > 0 AND < OR = TO 1.E+10 */
             TERM =1.E0,.
S10..
             IF ZZ LE 18.E0           /* ZZ < OR = 18                */
             THEN DO,.                /* TRANSLATE ARGUMENT          */
                  TERM =TERM*ZZ,.
                  ZZ   =ZZ+1.E0,.
                  GO TO S10,.
                  END,.
             ELSE DO,.                /* CALC. EQUATION 1            */
                  RZ2  =1.E0/ZZ**2,.
                  DLNG =(ZZ-C.5EC)*LOG(ZZ)-ZZ+0.918938533204672E0
                        -LOG(TERM)+(1.E0/ZZ)*(.8333333333333333E-01
                        -(RZ2*(.2777777777777777E-02+(RZ2*
                        (.79365C79365C793E-03-(RZ2*
                        (.595238C95238C95E-03))))))),.
                  GO TO S2C,.
                  END,.
             END,.
   ELSE IF XX LT 1.E7C                /* XX > 1.E+10 AND < 1.E+70    */
        THEN DO,.
             DLNG =ZZ*(LOG(ZZ)-1.EC),./* CALC. EQUATION 2            */
             GO TO S2C,.
             END,.
        ELSE DO,.                     /* XX > OR = 1.E+70            */
             ERROR='1',.              /* SET ERROR INDICATOR         */
             DLNG =1.E75,.
             END,.
S2C..
   RETURN,.
   END,.                              /* END OF PROCEDURE LGAM       */
```

Purpose:

LGAM computes the double-precision natural logarithm of the gamma function of a given double-precision argument.

Usage:

CALL LGAM (XX, DLNG);

XX - BINARY FLOAT (53)
Given double-precision argument for the log gamma function.

DLNG - BINARY FLOAT (53)
Resultant double-precision variable containing the log gamma function.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - If XX is greater than or equal to 10^{70}. If this condition exists, the value of DLNG is set to 1. E75.

ERROR=2 - If XX is less than or equal to 10^{-9}, DLNG is set to -1. E75.

Method:

For reference see:

M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, U. S. Department of Commerce, National Bureau of Standards Applied Mathematics Series, 1966, equation 6.1.4.

Mathematical Background:

This subroutine computes the double-precision natural logarithm of the gamma function of a given double-precision argument, xx, where $10^{-9} < xx < 10^{70}$. The Euler-McLaurin expansion, to the seventh derivative term, is used. For xx>0:

$$\log \Gamma(xx) = (xx - 1/2) \log xx$$

$$+ 1/2 \log 2\pi - xx + 1/(12\ xx) - 1/360\ xx^3$$

$$+ 1/1260\ xx^5 = 1/1680\ xx^7 \qquad (1)$$

This expression is very accurate for xx>18. If $xx \leq 18$, xx is replaced by $z = k + xx$, where k is an integer such that $z > 18$. Log Γ (z) is then evaluated by (1), and $\log xx + \log (xx + 1) + \ldots + \log (xx + k - 1)$ is subtracted to obtain the desired result.

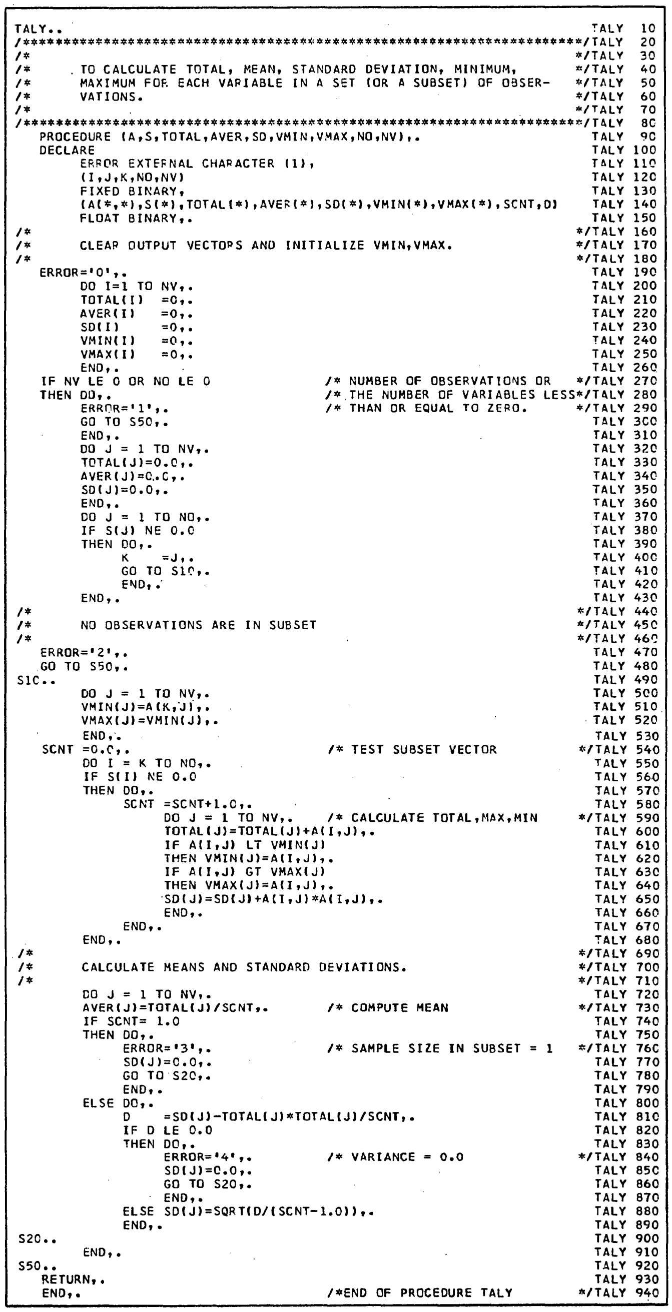
If xx is between 10^{10} and 10^{70}, terms of lowest order in (1) are neglected, and log Γ (xx) is computed as:

$$\log \Gamma(xx) = xx\ (\log(xx) - 1) \qquad (2)$$

Subroutine LGAM is available in a double-precision format only. If the single-precision value of the log gamma function of a given single-precision argument is desired, subroutine LGAM should be changed to single precision.

Data Screening and Analysis

• Subroutine TALY



```
TALY..
/**********************************************************************/
/*                                                                    */
/*       TO CALCULATE TOTAL, MEAN, STANDARD DEVIATION, MINIMUM,       */
/*       MAXIMUM FOR EACH VARIABLE IN A SET (OR A SUBSET) OF OBSER-   */
/*       VATIONS.                                                     */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,S,TOTAL,AVER,SD,VMIN,VMAX,NO,NV),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (I,J,K,NO,NV)
        FIXED BINARY,
        (A(*,*),S(*),TOTAL(*),AVER(*),SD(*),VMIN(*),VMAX(*),SCNT,D)
        FLOAT BINARY,.
/*                                                                    */
/*       CLEAR OUTPUT VECTORS AND INITIALIZE VMIN,VMAX.               */
/*                                                                    */
   ERROR='0',.
        DO I=1 TO NV,.
        TOTAL(I)  =0,.
        AVER(I)   =0,.
        SD(I)     =0,.
        VMIN(I)   =0,.
        VMAX(I)   =0,.
        END,.
   IF NV LE 0 OR NO LE 0                /* NUMBER OF OBSERVATIONS OR   */
   THEN DO,.                            /* THE NUMBER OF VARIABLES LESS*/
        ERROR='1',.                     /* THAN OR EQUAL TO ZERO.      */
        GO TO S50,.
        END,.
        DO J = 1 TO NV,.
        TOTAL(J)=0.0,.
        AVER(J)=0.0,.
        SD(J)=0.0,.
        END,.
        DO J = 1 TO NO,.
        IF S(J) NE 0.0
        THEN DO,.
             K    =J,.
             GO TO S10,.
             END,.
        END,.
/*                                                                    */
/*       NO OBSERVATIONS ARE IN SUBSET                                */
/*                                                                    */
   ERROR='2',.
   GO TO S50,.
S10..
        DO J = 1 TO NV,.
        VMIN(J)=A(K,J),.
        VMAX(J)=VMIN(J),.
        END,.
   SCNT =0.0,.                          /* TEST SUBSET VECTOR          */
        DO I = K TO NO,.
        IF S(I) NE 0.0
        THEN DO,.
             SCNT =SCNT+1.0,.
                  DO J = 1 TO NV,.    /* CALCULATE TOTAL,MAX,MIN     */
                  TOTAL(J)=TOTAL(J)+A(I,J),.
                  IF A(I,J) LT VMIN(J)
                  THEN VMIN(J)=A(I,J),.
                  IF A(I,J) GT VMAX(J)
                  THEN VMAX(J)=A(I,J),.
                  SD(J)=SD(J)+A(I,J)*A(I,J),.
                  END,.
             END,.
        END,.
/*                                                                    */
/*       CALCULATE MEANS AND STANDARD DEVIATIONS.                     */
/*                                                                    */
        DO J = 1 TO NV,.
        AVER(J)=TOTAL(J)/SCNT,.         /* COMPUTE MEAN                */
        IF SCNT= 1.0
        THEN DO,.
             ERROR='3',.                /* SAMPLE SIZE IN SUBSET = 1   */
             SD(J)=0.0,.
             GO TO S20,.
             END,.
        ELSE DO,.
             D    =SD(J)-TOTAL(J)*TOTAL(J)/SCNT,.
             IF D LE 0.0
             THEN DO,.
                  ERROR='4',.           /* VARIANCE = 0.0              */
                  SD(J)=0.0,.
                  GO TO S20,.
                  END,.
             ELSE SD(J)=SQRT(D/(SCNT-1.0)),.
             END,.
S20..
        END,.
S50..
   RETURN,.
   END,.                                /*END OF PROCEDURE TALY        */
```

Purpose:

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

Usage:

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

Description of parameters:

A(NO, NV) - BINARY FLOAT
Given observation matrix.

S(NO) - BINARY FLOAT
Given vector indicating subset of A. Only those observations with a nonzero S(J) are considered.

TOTAL(NV) - BINARY FLOAT
Resultant vector of totals.

AVER(NV) - BINARY FLOAT
Resultant vector of means.

SD(NV) - BINARY FLOAT
Resultant vector of standard deviations.

VMIN(NV) - BINARY FLOAT
Resultant vector of minima.

VMAX(NV) - BINARY FLOAT
Resultant vector of maxima.

NO - BINARY FIXED
Given parameter equal to the number of observations.

NV - BINARY FIXED
Given parameter equal to the number of variables.

Remarks:

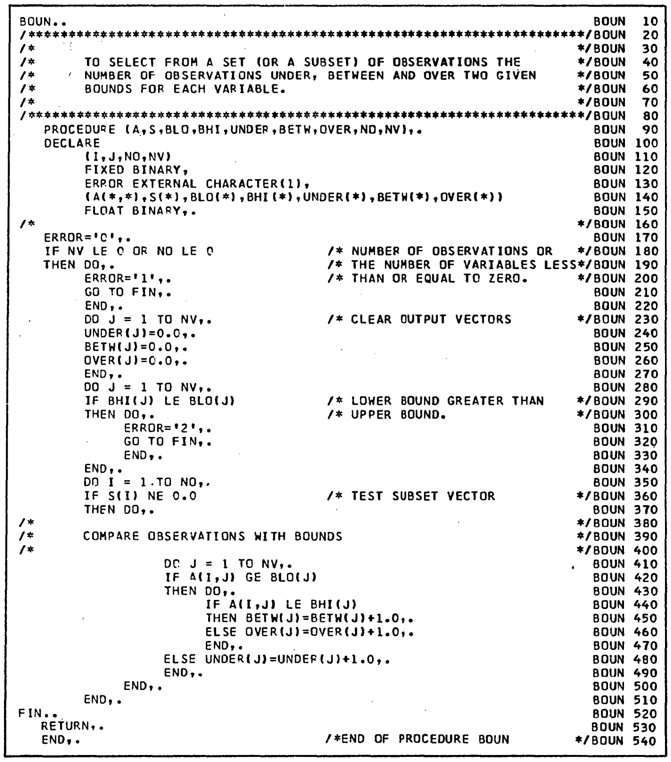
If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations or the number of variables less than or equal to zero.
ERROR=2 - no observations in subset vector.
ERROR=3 - sample size in subset equal to one.
ERROR=4 - variance equal to zero.

Method:

All observations corresponding to a nonzero element in the 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 deviations is one less than the number of observations used.

- <u>Subroutine</u> BOUN



```
BOUN..
/**********************************************************************/
/*                                                                    */
/*      TO SELECT FROM A SET (OR A SUBSET) OF OBSERVATIONS THE       */
/*      NUMBER OF OBSERVATIONS UNDER, BETWEEN AND OVER TWO GIVEN   */
/*      BOUNDS FOR EACH VARIABLE.                                     */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,S,BLO,BHI,UNDER,BETW,OVER,NO,NV),.
   DECLARE
        (I,J,NO,NV)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (A(*,*),S(*),BLO(*),BHI(*),UNDER(*),BETW(*),OVER(*))
        FLOAT BINARY,.
/*                                                                    */
   ERROR='0',.
   IF NV LE 0 OR NO LE 0              /* NUMBER OF OBSERVATIONS OR    */
   THEN DO,.                          /* THE NUMBER OF VARIABLES LESS*/
        ERROR='1',.                   /* THAN OR EQUAL TO ZERO.       */
        GO TO FIN,.
        END,.
        DO J = 1 TO NV,.              /* CLEAR OUTPUT VECTORS         */
        UNDER(J)=0.0,.
        BETW(J)=0.0,.
        OVER(J)=0.0,.
        END,.
        DO J = 1 TO NV,.
        IF BHI(J) LE BLO(J)           /* LOWER BOUND GREATER THAN     */
        THEN DO,.                     /* UPPER BOUND.                 */
             ERROR='2',.
             GO TO FIN,.
             END,.
        END,.
        DO I = 1.TO NO,.
        IF S(I) NE 0.0                /* TEST SUBSET VECTOR           */
        THEN DO,.
/*                                                                    */
/*      COMPARE OBSERVATIONS WITH BOUNDS                              */
/*                                                                    */
                  DO J = 1 TO NV,.
                  IF A(I,J) GE BLO(J)
                  THEN DO,.
                       IF A(I,J) LE BHI(J)
                       THEN BETW(J)=BETW(J)+1.0,.
                       ELSE OVER(J)=OVER(J)+1.0,.
                       END,.
                  ELSE UNDER(J)=UNDER(J)+1.0,.
                  END,.
             END,.
        END,.
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE BOUN         */
```

Purpose:

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

Usage:

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

Description of parameters:

A(NO, NV) - BINARY FLOAT
Given observation matrix.

S(NO) - BINARY FLOAT
Given vector indicating subset of A. Only those observations with a nonzero S(J) are considered.

BLO(NV) - BINARY FLOAT
Given vector of lower bounds on all variables.

BHI(NV) - BINARY FLOAT
Given vector of upper bounds on all variables.

UNDER(NV) - BINARY FLOAT
Resultant vector indicating, for each variable, number of observations under lower bounds.

BETW(NV) - BINARY FLOAT
Resultant vector indicating, for each variable, number of observations equal to or between lower and upper bounds.

OVER(NV) - BINARY FLOAT
Resultant vector indicating, for each variable, number of observations over upper bounds.

NO - BINARY FIXED
Given number of observations.

NV - BINARY FIXED
Given number of variables for each observation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations or number of variables less than or equal to zero.
ERROR=2 - lower bound greater than upper bound.

Method:

Each row (observation) of the matrix A with corresponding nonzero element in S vector is tested. Observations are compared with specified lower and upper variable bounds and counts are kept in vectors UNDER, BETW and OVER.

- <u>Subroutine ABST</u>

```
ABST..
/**********************************************************************/
/*                                                                    */
/*      TO TEST MISSING OR ZERO VALUES FOR OBSERVATION MATRIX A.     */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,S,NO,NV),.
   DECLARE
        (I,J,NO,NV)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (A(*,*),S(*)) FLOAT BINARY,.
/*                                                                    */
   ERROR='0',.
   IF NV LE 0 OR NO LE 0               /* NUMBER OF OBSERVATIONS OR   */
   THEN DO,.                           /* THE NUMBER OF VARIABLES LESS*/
        ERROR='1',.                    /* THAN OR EQUAL TO ZERO.      */
        GO TO FIN,.
        END,.
        DO I = 1 TO NO,.
             DO J = 1 TO NV,.
             IF A(I,J)= 0.0
             THEN DO,.
                  S(I) =0.0,.
                  GO TO S10,.
                  END,.
             END,.
        S(I) =1.0,.
S10..
        END,.
FIN..
   RETURN,.
   END,.                               /*END OF PROCEDURE ABST        */
```

Purpose:

ABST tests for missing or zero elements in observation matrix A.

Usage:

CALL ABST (A, S, NO, NV);

Description of parameters:

A(NO, NV) - BINARY FLOAT
Given observation matrix.

S(NO) - BINARY FLOAT
Resultant vector indicating one of the following codes for each observation:
1 There is not a missing or zero value.
0 At least one variable has a value missing or zero.

NO - BINARY FIXED
Given number of observations.

NV - BINARY FIXED
Given number of variables for each observation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR=1 - number of observations or number of variables less than or equal to zero.

Method:

A test is made on the I-th row (observation) of the matrix A, I = 1, ..., NO. If there is not a missing or zero value, 1 is placed in S(I). If at least one variable has a value missing or zero, 0 is placed in S(I).

● Subroutine SBST

```
SBST..
/**********************************************************************/
/*                                                                    */
/*      TO DERIVE A SUBSET VECTOR INDICATING WHICH OBSERVATIONS IN    */
/*      A SET HAVE SATISFIED CERTAIN CONDITIONS.                      */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,C,R,B,S,NO,NV,NC),.
   DECLARE
         B ENTRY,
         ERROR EXTERNAL CHARACTER(1),
         (I,ICOL,IGO,J,NC,NO)
         FIXED BINARY,
         (A(*,*),C(*,*),R(*),S(*),Q,TR)
         BINARY FLOAT,
         T(6) LABEL,.
/*                                                                    */
   ERROR='0',.
         DO I=1 TO NO,.
         S(I) =0,.
         END,.
   IF NO LE 0 OF NV LE 0 OR NC LE 0   /* NUMBER OF OBSERVATIONS,      */
   THEN DO,.                          /* VARIABLES, OR CONDITIONS IS  */
         ERROR='1',.                  /* LESS THAN OR EQUAL TO ZERO.  */
         GO TO FIN,.
         END,.
         DO I = 1 TO NO,.
               DO J = 1 TO NC,.
               R(J) =0.0,.            /* CLEAR R VECTOR               */
/*                                                                    */
/*      LOCATE ELEMENT IN OBSERVATION MATRIX AND RELATION CODE        */
/*                                                                    */
               ICOL =C(1,J),.
               IGO  =C(2,J),.
               IF IGO LT 1 OR IGO GT 6 /* CONDITION VALUE INVALID     */
               THEN DO,.
                     ERROR='2',.
                     GO TO FIN,.
                     END,.
               IF ICOL LT 1 OR ICOL GT NV
               THEN DO,.
                     ERROR ='3',.     /* INVALID VARIABLE NUMBER      */
                     GO TO FIN,.
                     END,.
               Q    =A(I,ICOL)-C(3,J),. /* FORM R VECTOR              */
               GO TO T(IGO),.
T(1)..
               IF Q LT 0.0
               THEN GO TO S10,.
               GO TO S20,.
T(2)..
               IF Q LE 0.0
               THEN GO TO S10,.
               GO TO S20,.
T(3)..
               IF Q = 0.0
               THEN GO TO S10,.
               GO TO S20,.
T(4)..
               IF Q NE 0.0
               THEN GO TO S10,.
               GO TO S20,.
T(5)..
               IF Q GE 0.0
               THEN GO TO S10,.
               GO TO S20,.
T(6)..
               IF Q LE 0.0
               THEN GO TO S20,.
S10..
               R(J) =1.0,.
S20..
               END,.
         CALL B (R,TR),.              /* CALCULATE S VECTOR           */
         S(I) =TR,.
         END,.
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE SBST         */
```

Purpose:

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

Usage:

CALL SBST (A, C, R, B, S, NO, NV, NC);
Parameter B must be declared as an entry attribute in the calling program.

A(NO, NV) - BINARY FLOAT
Given observation matrix.

C(3, NC) - BINARY FLOAT
Given matrix of conditions to be considered. The first element of each column of C represents the number of the variable (column of matrix A) to be tested. The second element of each column is a relation code as follows:

1 - less than
2 - less than or equal to
3 - equal to
4 - not equal to
5 - greater than or equal to
6 - 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(NC) - BINARY FLOAT
Resultant 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.

B - ENTRY
Given 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

```
BOOL. .
   PROCEDURE (R, T), .
      DECLARE
      (R(*), T)
      FLOAT BINARY, .
   T=R(1)*R(2), .
   RETURN, .
   END, .
```

The above tests for R(1) and R(2).

S(NO) - BINARY FLOAT
Resultant vector indicating, for each observation, whether or not proposition B is satisfied. If it is, S(I) is nonzero. If it is not, S(I) is zero.

NO - BINARY FIXED
Given number of observations.

NV - BINARY FIXED
Given number of variables.

NC - BINARY FIXED
Given number of basic conditions to be satisfied.

Remarks:

Subroutines and function subroutines required:

B - The name of the actual subroutine supplied by the user may be different from B (for example, BOOL), but subroutine SBST always calls B. In order for procedure SBST to do this, the name of the user-supplied procedure must be defined by an entry attribute in the calling program.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations, number of variables, or number of conditions is less than or equal to zero.
ERROR=2 - condition value invalid.
ERROR=3 - variable number is less than 1 or greater than the number of variables.

Method:

The following is done for each observation. Condition matrix is analyzed to determine which variables are to be examined. The 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 TAB1

```
TAB1..
/*********************************************************************/
/*                                                                   */
/*      TO TABULATE FOR ONE VARIABLE IN AN OBSERVATION MATRIX (OR A  */
/*      SUBSET), THE FREQUENCY AND PEPCENT FREQUENCY OVER GIVEN      */
/*      CLASS INTERVALS. IN ADDITION, CALCULATE FOR THE SAME VARIABLE*/
/*      THE TOTAL, MEAN, STANDARD DEVIATION, MINIMUM, AND            */
/*      MAXIMUM.                                                     */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,S,NOVAR,UBO,FREQ,PCT,STATS,NO,NV),.
   DECLARE
        EPFOR EXTERNAL CHAPACTER (1),
        (I,INN,INTX,J,K,NO,NOVAR,KK)
        FIXED BINARY,
        (A(*,*),S(*),UBO(*),FREQ(*),PCT(*),STATS(*),SCNT,VMIN,VMAX,
        SINT,TEMP)
        BINARY FLOAT,.
/*                                                                   */
   ERROR='C',.
        IF NOVAP LE 0 OR NOVAF GT NV  /* VALUE OF THE VARIABLE TO BE */
        THEN DO,.                     /* TABULATED IS INVALID        */
             ERROR='6',.
             GO TO S50,.
             END,.
   IF NV LE 0 OR NO LE 0
   THEN DO,.                          /* NUMBER OF OBSERVATIONS OR   */
        ERROR='1',.                   /* THE NUMBER OF VARIABLES ARE */
        GO TO S50,.                   /* LESS THAN OR EQUAL TO ZERO. */
        END,.
   INN  =UBO(2),.                     /* CALCULATE INTERVAL SIZE     */
        DO J = 1 TO INN,.             /* CLEAR OUTPUT VECTORS        */
        FREQ(J)=0.C,.
        PCT(J)=0.0,.
        END,.
        DO J = 1 TO 5,.
        STATS(J)=C.0,.
        END,.
   IF UBC(1) GT UBO(3) OR UBO(2) LE 2.0
   THEN DO,.                          /* INVALID BOUNDS OR THE NUMBER*/
        ERROR='2',.                   /* OF INTERVALS LESS THAN OR   */
        GO TO S5C,.                   /* EQUAL TO TWO.               */
        END,.
        DO I = 1 TO NO,.              /* CALCULATE MAX AND MIN       */
        IF S(I) NE 0.C
        THEN DO,.
             KK   =I,.
             VMIN =A(I,NOVAR),.
             VMAX =VMIN,.
             GO TO S10,.
             END,.
        END,.
   ERROR='3',.                        /* NO OBSERVATION IN SUBSET    */
   GO TO S5C,.
S10..
        DO I = KK TO NO,.
        IF S(I) NE 0.0
        THEN DO,.
             IF A(I,NOVAR) LT VMIN
             THEN VMIN =A(I,NOVAR),.
             IF A(I,NOVAR) GT VMAX
             THEN VMAX =A(I,NOVAR),.
             END,.
        END,.
   STATS(4)=VMIN,.
   STATS(5)=VMAX,.
   IF UBO(1)= UBO(3)
   THEN DO,.
        UBO(1)=VMIN,.
        UBO(3)=VMAX,.
        END,.
   SINT =(UBO(3)-UBO(1))/(UBO(2)-2),.
   SCNT =0.0,.                        /* TEST SUBSET VECTOR          */
        DO I = KK TO NO,.
        IF S(I) NE 0.0
        THEN DO,.
             SCNT =SCNT+1.0,.
/*                                                                   */
/*      DEVELOP TOTALS AND FREQUENCIES                               */
/*                                                                   */
             STATS(1)=STATS(1)+A(I,NOVAR),.
             STATS(3)=STATS(3)+A(I,NOVAR)**2,.
             TEMP =UBO(1)-SINT,.
             INTX =INN-1,.
                  DO J = 1 TO INTX,.
                  TEMP =TEMP+SINT,.
                  IF A(I,NOVAR) LT TEMP
                  THEN DO,.
                       K    =J,.
                       GO TO S2C,.
                       END,.
                  END,.
             IF A(I,NOVAR) GE TEMP
             THEN DO,.
                  FREQ(INN)=FREQ(INN)+1.0,.
                  GO TO S30,.
                  END,.
S20..
             FREQ(K)=FREQ(K)+1.0,.
             END,.
S30..
        END,.
/*                                                                   */
/*      CALCULATE RELATIVE FREQUENCIES                               */
/*                                                                   */
        DO J = 1 TO INN,.
        PCT(J)=FREQ(J)*100.0/SCNT,.
        END,.
/*                                                                   */
/*      CALCULATE MEAN AND STANDARD DEVIATION                        */
/*                                                                   */
   STATS(2)=STATS(1)/SCNT,.
   IF SCNT= 1.0
   THEN DO,.
        ERROR='4',.                   /* SAMPLE SIZE = 1             */
        STATS(3)=0.C
        GO TO S50,.
        END,.
   ELSE DO,.
        TEMP =STATS(3)-STATS(1)*STATS(1)/SCNT,.
        IF TEMP LE 0.0
        THEN DO,.
             ERROR='5',.              /* VARIANCE = 0.C              */
```

```
          STATS(3)=0.0,.                                          TAB11240
          GO TO S50,.                                             TAB11250
          END,.                                                   TAB11260
      ELSE STATS(3)=SQPT(TEMP/(SCNT-1.0)),.                       TAB11270
      END,.                                                       TAB11280
S50..                                                             TAB11290
   RETURN,.                                                       TAB11300
   END,.                                /*END OF PROCEDURE TAB1 */TAB11310
```

Purpose:

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

Usage:

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

Description of parameters:

A(NO, NV) - BINARY FLOAT
Given observation matrix A.

S(NO) - BINARY FLOAT
Given vector that indicates which of the observations enter the calculation. A zero element in S indicates that the corresponding observation of A is not to be included.

NOVAR - BINARY FIXED
Given variable to be tabulated.

UBO(3) - BINARY FLOAT
Given vector containing 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 replaces these with the minimum and maximum values of the variable. Number of intervals, UBO(2), must include two cells for values under and above limits.

FREQ (INN) - BINARY FLOAT
Resultant vector of frequencies. INN is given in UBO(2).

PCT(INN) - BINARY FLOAT
Resultant vector of relative frequencies. Vector length is UBO(2).

STATS(5) - BINARY FLOAT
Resultant vector of summary statistics, that is, total, mean, standard deviation, minimum, and maximum.

NO - BINARY FIXED
Given number of observations.

NV - BINARY FIXED
Given number of variables for each observation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations or number of variables less than or equal to zero.
ERROR=2 - invalid bounds or number of intervals less than or equal to two.
ERROR=3 - no observations in subset.
ERROR=4 - sample size equal to one.
ERROR=5 - variance equal to zero.
ERROR=6 - value of the variable to be tabulated is invalid.

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.

Mathematical Background:

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:

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

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, F_i, of the i^{th} class interval for the variable, where $i = 1, 2, \ldots, UBO_2$. Then each frequency is divided by the number of observations, n, to obtain the percent frequency:

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

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

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

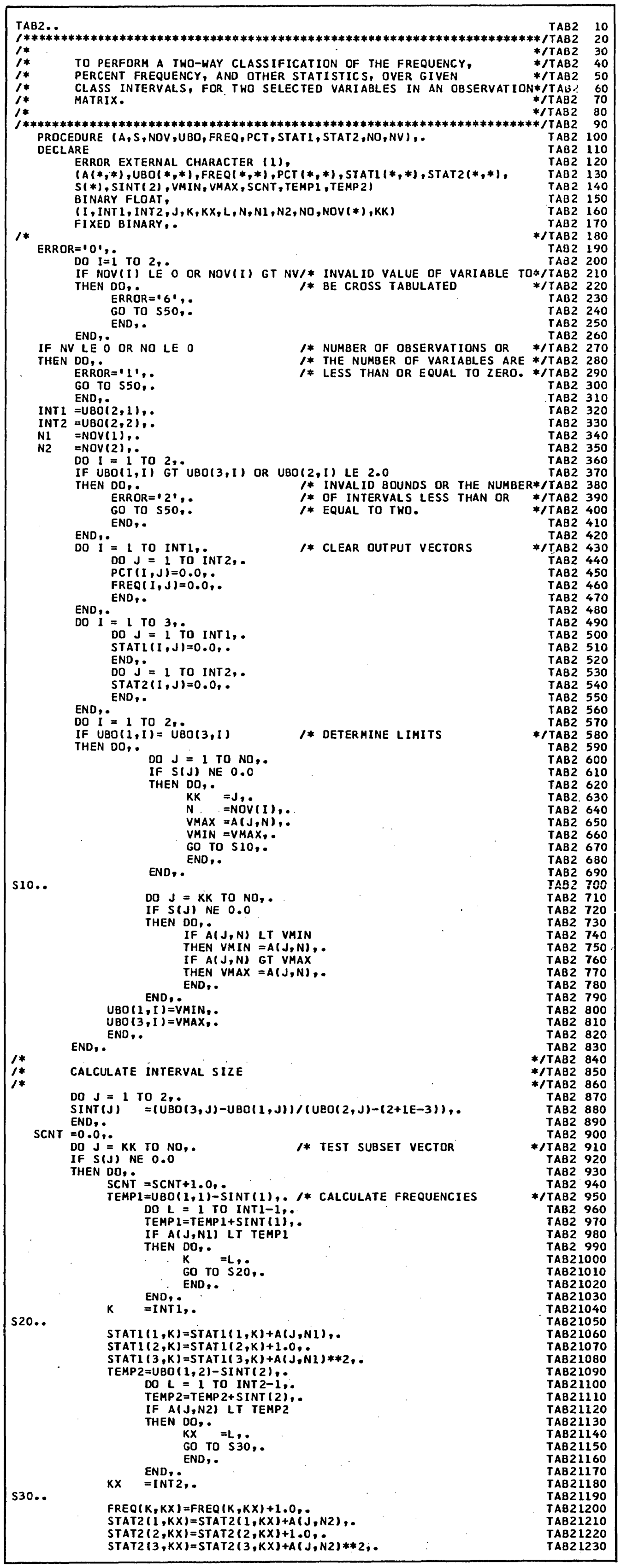
where j = selected variable

Mean: $$\overline{X} = \frac{T}{n} \qquad (4)$$

Standard deviation:

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

- Subroutine TAB2



```
TAB2..
/*********************************************************************/
/*                                                                  */
/*       TO PERFORM A TWO-WAY CLASSIFICATION OF THE FREQUENCY,      */
/*       PERCENT FREQUENCY, AND OTHER STATISTICS, OVER GIVEN        */
/*       CLASS INTERVALS, FOR TWO SELECTED VARIABLES IN AN OBSERVATION*/
/*       MATRIX.                                                    */
/*                                                                  */
/*********************************************************************/
   PROCEDURE (A,S,NOV,UBO,FREQ,PCT,STAT1,STAT2,NO,NV),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (A(*,*),UBO(*,*),FREQ(*,*),PCT(*,*),STAT1(*,*),STAT2(*,*),
        S(*),SINT(2),VMIN,VMAX,SCNT,TEMP1,TEMP2)
        BINARY FLOAT,
        (I,INT1,INT2,J,K,KX,L,N,N1,N2,NO,NOV(*),KK)
        FIXED BINARY,.
/*                                                                  */
   ERROR='0',.
        DO I=1 TO 2,.
        IF NOV(I) LE 0 OR NOV(I) GT NV/* INVALID VALUE OF VARIABLE TO*/
        THEN DO,.                     /* BE CROSS TABULATED         */
             ERROR='6',.
             GO TO S50,.
             END,.
        END,.
   IF NV LE 0 OR NO LE 0              /* NUMBER OF OBSERVATIONS OR  */
   THEN DO,.                          /* THE NUMBER OF VARIABLES ARE */
        ERROR='1',.                   /* LESS THAN OR EQUAL TO ZERO. */
        GO TO S50,.
        END,.
   INT1 =UBO(2,1),.
   INT2 =UBO(2,2),.
   N1   =NOV(1),.
   N2   =NOV(2),.
        DO I = 1 TO 2,.
        IF UBO(1,I) GT UBO(3,I) OR UBO(2,I) LE 2.0
        THEN DO,.                     /* INVALID BOUNDS OR THE NUMBER*/
             ERROR='2',.              /* OF INTERVALS LESS THAN OR  */
             GO TO S50,.              /* EQUAL TO TWO.              */
             END,.
        END,.
        DO I = 1 TO INT1,.            /* CLEAR OUTPUT VECTORS       */
             DO J = 1 TO INT2,.
             PCT(I,J)=0.0,.
             FREQ(I,J)=0.0,.
             END,.
        END,.
        DO I = 1 TO 3,.
             DO J = 1 TO INT1,.
             STAT1(I,J)=0.0,.
             END,.
             DO J = 1 TO INT2,.
             STAT2(I,J)=0.0,.
             END,.
        END,.
        DO I = 1 TO 2,.
        IF UBO(1,I)= UBO(3,I)         /* DETERMINE LIMITS           */
        THEN DO,.
                  DO J = 1 TO NO,.
                  IF S(J) NE 0.0
                  THEN DO,.
                       KK   =J,.
                       N    =NOV(I),.
                       VMAX =A(J,N),.
                       VMIN =VMAX,.
                       GO TO S10,.
                       END,.
                  END,.
S10..
                  DO J = KK TO NO,.
                  IF S(J) NE 0.0
                  THEN DO,.
                       IF A(J,N) LT VMIN
                       THEN VMIN =A(J,N),.
                       IF A(J,N) GT VMAX
                       THEN VMAX =A(J,N),.
                       END,.
                  END,.
             UBO(1,I)=VMIN,.
             UBO(3,I)=VMAX,.
             END,.
        END,.
/*                                                                  */
/*      CALCULATE INTERVAL SIZE                                     */
/*                                                                  */
        DO J = 1 TO 2,.
        SINT(J)    =(UBO(3,J)-UBO(1,J))/(UBO(2,J)-(2+1E-3)),.
        END,.
   SCNT =0.0,.
        DO J = KK TO NO,.             /* TEST SUBSET VECTOR         */
        IF S(J) NE 0.0
        THEN DO,.
             SCNT =SCNT+1.0,.
             TEMP1=UBO(1,1)-SINT(1),. /* CALCULATE FREQUENCIES      */
                  DO L = 1 TO INT1-1,.
                  TEMP1=TEMP1+SINT(1),.
                  IF A(J,N1) LT TEMP1
                  THEN DO,.
                       K    =L,.
                       GO TO S20,.
                       END,.
                  END,.
             K    =INT1,.
S20..
             STAT1(1,K)=STAT1(1,K)+A(J,N1),.
             STAT1(2,K)=STAT1(2,K)+1.0,.
             STAT1(3,K)=STAT1(3,K)+A(J,N1)**2,.
             TEMP2=UBO(1,2)-SINT(2),.
                  DO L = 1 TO INT2-1,.
                  TEMP2=TEMP2+SINT(2),.
                  IF A(J,N2) LT TEMP2
                  THEN DO,.
                       KX   =L,.
                       GO TO S30,.
                       END,.
                  END,.
             KX   =INT2,.
S30..
             FREQ(K,KX)=FREQ(K,KX)+1.0,.
             STAT2(1,KX)=STAT2(1,KX)+A(J,N2),.
             STAT2(2,KX)=STAT2(2,KX)+1.0,.
             STAT2(3,KX)=STAT2(3,KX)+A(J,N2)**2,.
```

```
          END,.                                                         TAB21240
       END,.                                                            TAB21250
  IF SCNT= 0.0                                                          TAB21260
  THEN DO,.                                                             TAB21270
       ERROR='3',.                  /* NO OBSERVATIONS IN SUBSET       */TAB21280
       GO TO S50,.                                                      TAB21290
       END,.                                                            TAB21300
/*                                                                     */TAB21310
/*     CALCULATE PERCENT FREQUENCIES.                                  */TAB21320
/*                                                                     */TAB21330
       DO I = 1 TO INT1,.                                               TAB21340
          DO J = 1 TO INT2,.                                            TAB21350
          PCT(I,J)=FREQ(I,J)*100.0/SCNT,.                               TAB21360
          END,.                                                         TAB21370
       END,.                                                            TAB21380
/*                                                                     */TAB21390
/*     CALCULATE TOTALS, MEANS, STANDARD DEVIATIONS                    */TAB21400
/*                                                                     */TAB21410
       DO J = 1 TO INT1,.                                               TAB21420
       IF STAT1(2,J) LE 1.0                                             TAB21430
       THEN DO,.                                                        TAB21440
          ERROR='4',.               /* NUMBER OF OBSERVATIONS IS       */TAB21450
          STAT1(3,J)=0.0,.          /* LESS THAN OR EQUAL TO 1 IN      */TAB21460
          STAT1(2,J)=STAT1(1,J),.   /* SOME INTERVAL                   */TAB21470
          END,.                                                         TAB21480
       ELSE DO,.                                                        TAB21490
          TEMP1=STAT1(3,J)-STAT1(1,J)**2/STAT1(2,J),.                   TAB21500
          STAT1(2,J)=STAT1(1,J)/STAT1(2,J),.                            TAB21510
          IF TEMP1 LE 0.0                                               TAB21520
          THEN DO,.                                                     TAB21530
             ERROR='5',.            /* VARIANCE IS 0.0                 */TAB21540
             STAT1(3,J)=0.0,.                                           TAB21550
             END,.                                                      TAB21560
          ELSE STAT1(3,J)=SQRT(TEMP1/(STAT1(2,J)-1.0)),.                TAB21570
          END,.                                                         TAB21580
       END,.                                                            TAB21590
/*                                                                     */TAB21600
       DO J = 1 TO INT2,.                                               TAB21610
       IF STAT2(2,J) LE 1.0                                             TAB21620
       THEN DO,.                                                        TAB21630
          ERROR='4',.               /* NUMBER OF OBSERVATIONS IS       */TAB21640
          STAT2(3,J)=0.0,.          /* LESS THAN OR EQUAL TO 1 IN      */TAB21650
          STAT2(2,J)=STAT2(1,J),.   /* SOME INTERVAL                   */TAB21660
          END,.                                                         TAB21670
       ELSE DO,.                                                        TAB21680
          STAT2(2,J)=STAT2(1,J)/STAT2(2,J),.                            TAB21690
          TEMP2=STAT2(3,J)-STAT2(1,J)**2/STAT2(2,J),.                   TAB21700
          IF TEMP2 LE 0.0                                               TAB21710
          THEN DO,.                                                     TAB21720
             ERROR='5',.            /* VARIANCE = 0.0                  */TAB21730
             STAT2(3,J)=0.0,.                                           TAB21740
             END,.                                                      TAB21750
          ELSE STAT2(3,J)=SQRT(TEMP2/(STAT2(2,J)-1.0)),.                TAB21760
          END,.                                                         TAB21770
       END,.                                                            TAB21780
S50..                                                                   TAB21790
   RETURN,.                                                             TAB21800
   END,.                            /*END OF PROCEDURE TAB2            */TAB21810
```

Purpose:

TAB2 performs 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(NO, NV) - BINARY FLOAT
Given observation matrix.

S(ND) - BINARY FLOAT
Given vector that indicates which of the observations enter the calculation. A zero element in S indicates that the corresponding observation of A is not to be included.

NOV(2) - BINARY FIXED
Given variables to be cross-tabulated. NOV(1) is variable 1; NOV(2) is variable 2.

UBO(3, 2) - BINARY FLOAT
Given 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 a variable, the program replaces these with the minimum and maximum values of that variable. Number of intervals must include two cells for under and above limits.

FREQ (INT1, INT2) - BINARY FLOAT
Resultant matrix of frequencies in the two-way classification. INT1 equals UBO(2, 1) and INT2 equals UBO(2, 2) where UBO(2, 1) is the number of intervals of variable 1 and UBO(2, 2) is the number of intervals of variable 2. UBO(2, 1) and UBO(2, 2) must be specified in the second position of the respective column of UBO matrix.

PCT (INT1,INT2) - BINARY FLOAT
Resultant matrix of percent frequencies.

STAT1 (3, INT2) BINARY FLOAT
Resultant matrix summarizing totals, means, and standard deviations for each class interval of variable 1.

STAT2 (3,INT2) -
Same as STAT1 but over variable 2.

NO - BINARY FIXED
Given number of observations.

NV - BINARY FIXED
Given number of variables for each observation.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations or number of variables less than or equal to zero.
ERROR=2 - invalid bounds or number of intervals less than or equal to two.
ERROR=3 - no observations in subset.
ERROR=4 - number of observations one or less in some interval.

ERROR=5 - variance equal to zero. (If error conditions 4 and 5 exist, the last condition encountered overrides.)

ERROR=6 - invalid value of variable to be cross-tabulated.

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.

Mathematical Background:

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:

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

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^{th} variable.

A frequency tabulation is then made for each pair of observations in a two-way table as shown in Figure 10.

Symbols $\geq$ and $<$ in Figure 10 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:

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

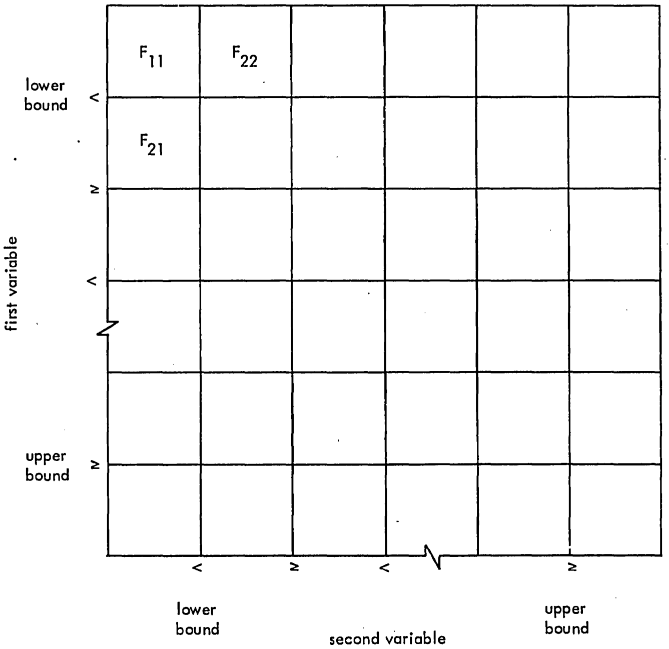


Figure 10. Frequency matrix

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

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

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

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

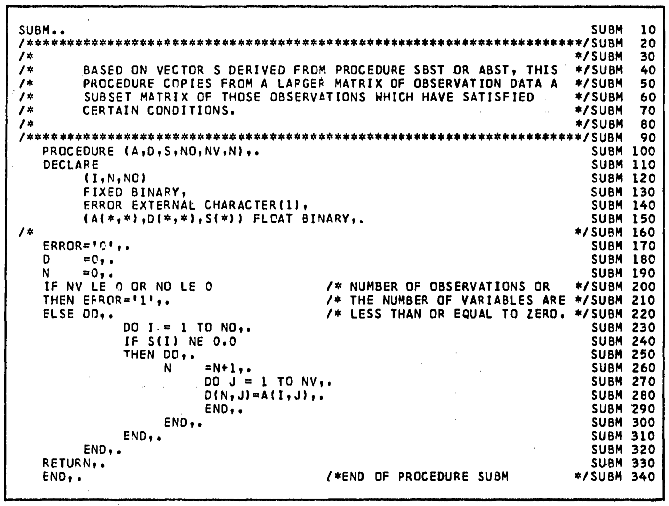
From these, the following statistics are calculated for each class interval:

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

Standard deviation:

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

• Subroutine SUBM



```
SUBM..
/*********************************************************************/
/*                                                                   */
/*      BASED ON VECTOR S DERIVED FROM PROCEDURE SBST OR ABST, THIS  */
/*      PROCEDURE COPIES FROM A LAPGER MATRIX OF OBSERVATION DATA A  */
/*      SUBSET MATRIX OF THOSE OBSERVATIONS WHICH HAVE SATISFIED     */
/*      CERTAIN CONDITIONS.                                          */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,D,S,NO,NV,N),.
   DECLARE
        (I,N,NO)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (A(*,*),D(*,*),S(*)) FLCAT BINARY,.
/*                                                                   */
   ERROR='0',.
   D    =0,.
   N    =0,.
   IF NV LE 0 OR NO LE 0                  /* NUMBER OF OBSERVATIONS OR   */
   THEN EFROR='1',.                       /* THE NUMBER OF VARIABLES ARE */
   ELSE DO,.                              /* LESS THAN OR EQUAL TO ZERO. */
            DO I = 1 TO NO,.
            IF S(I) NE 0.0
            THEN DO,.
                 N    =N+1,.
                      DO J = 1 TO NV,.
                      D(N,J)=A(I,J),.
                      END,.
                 END,.
            END,.
        END,.
   RETURN,.
   END,.                                  /*END OF PROCEDURE SUBM        */
```

Purpose:

SUBM copies a submatrix from an observation matrix. The elements of this submatrix satisfy conditions specified by an input vector. This subroutine is used in preparing data for input to a statistical analysis such as multiple regression.

Usage:

CALL SUBM (A, D, S, NO, NV, N);

Description of parameters:

A(NO,NV) - BINARY FLOAT
Given matrix of observations.

D(N, NV) - BINARY FLOAT
Resultant matrix of observations.

S(NO) - BINARY FLOAT
Given vector containing the codes derived from procedures SBST or ABST.

NO - BINARY FIXED
Given number of observations.

NV - BINARY FIXED
Given number of variables for each observation.

N - BINARY FIXED
Resultant variable containing the number of nonzero codes in vector S.

Remarks:

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

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero.

The following constitutes the possible error condition that may be detected:

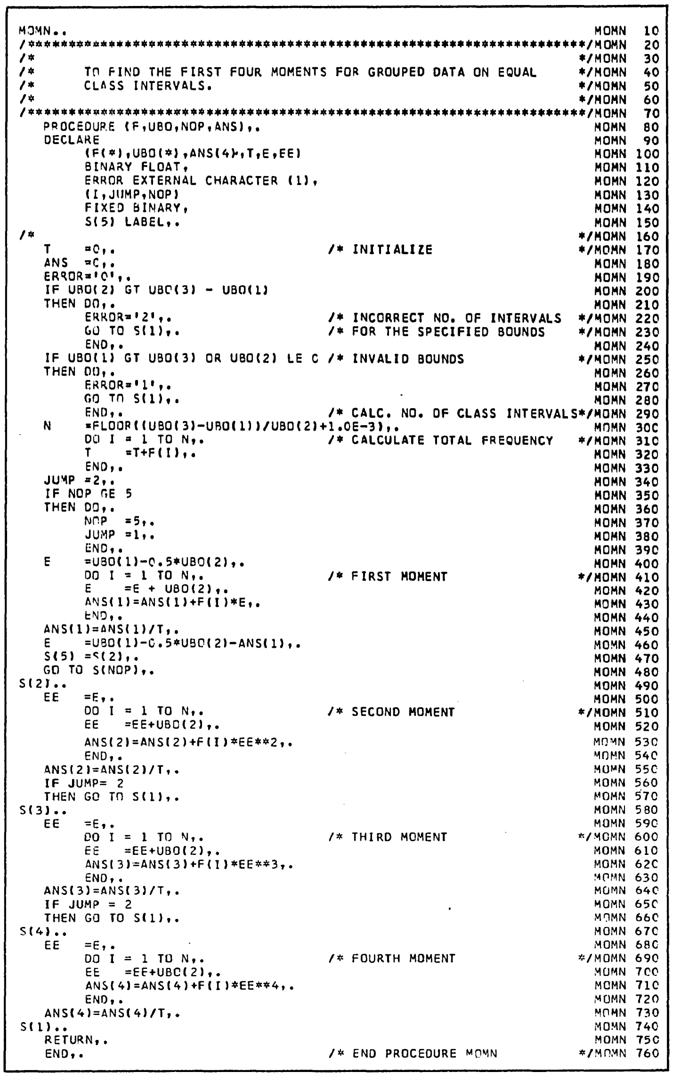
ERROR=1 - number of observations or number of variables less than or equal to zero.

Method:

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

Elementary Statistics

• Subroutine MOMN



```
MOMN..
/***********************************************************************/
/*                                                                     */
/*      TO FIND THE FIRST FOUR MOMENTS FOR GROUPED DATA ON EQUAL       */
/*      CLASS INTERVALS.                                               */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (F,UBO,NOP,ANS),.
   DECLARE
        (F(*),UBO(*),ANS(4),T,E,EE)
        BINARY FLOAT,
        ERROR EXTERNAL CHARACTER (1),
        (I,JUMP,NOP)
        FIXED BINARY,
        S(5) LABEL,.
/*                                                                     */
   T    =0,.                        /* INITIALIZE                      */
   ANS  =0,.
   ERROR='0',.
   IF UBO(2) GT UBO(3) - UBO(1)
   THEN DO,.
        ERROR='2',.                 /* INCORRECT NO. OF INTERVALS      */
        GO TO S(1),.                /* FOR THE SPECIFIED BOUNDS        */
        END,.
   IF UBO(1) GT UBO(3) OR UBO(2) LE 0 /* INVALID BOUNDS                */
   THEN DO,.
        ERROR='1',.
        GO TO S(1),.
        END,.                       /* CALC. NO. OF CLASS INTERVALS*/
   N    =FLOOR((UBO(3)-UBO(1))/UBO(2)+1.0E-3),.
        DO I = 1 TO N,.             /* CALCULATE TOTAL FREQUENCY       */
        T    =T+F(I),.
        END,.
   JUMP =2,.
   IF NOP GE 5
   THEN DO,.
        NOP  =5,.
        JUMP =1,.
        END,.
   E    =UBO(1)-0.5*UBO(2),.
        DO I = 1 TO N,.             /* FIRST MOMENT                    */
        E    =E + UBO(2),.
        ANS(1)=ANS(1)+F(I)*E,.
        END,.
   ANS(1)=ANS(1)/T,.
   E    =UBO(1)-0.5*UBO(2)-ANS(1),.
   S(5) =S(2),.
   GO TO S(NOP),.
S(2)..
   EE   =E,.
        DO I = 1 TO N,.             /* SECOND MOMENT                   */
        EE   =EE+UBO(2),.
        ANS(2)=ANS(2)+F(I)*EE**2,.
        END,.
   ANS(2)=ANS(2)/T,.
   IF JUMP= 2
   THEN GO TO S(1),.
S(3)..
   EE   =E,.
        DO I = 1 TO N,.             /* THIRD MOMENT                    */
        EE   =EE+UBO(2),.
        ANS(3)=ANS(3)+F(I)*EE**3,.
        END,.
   ANS(3)=ANS(3)/T,.
   IF JUMP = 2
   THEN GO TO S(1),.
S(4)..
   EE   =E,.
        DO I = 1 TO N,.             /* FOURTH MOMENT                   */
        EE   =EE+UBO(2),.
        ANS(4)=ANS(4)+F(I)*EE**4,.
        END,.
   ANS(4)=ANS(4)/T,.
S(1)..
   RETURN,.
   END,.                            /* END PROCEDURE MOMN              */
```

Purpose:

MOMN finds the first four moments for grouped data on equal class intervals.

Usage:

CALL MOMN (F, UBO, NOP, ANS);

F(N) - BINARY FLOAT
Given vector containing grouped data, (frequencies), where N is the number of class intervals.

UBO(3) - BINARY FLOAT
Given vector containing the lower bound, UBO(1), the class interval, UBO(2), and the upper limit, UBO(3).

NOP - BINARY FIXED
Given option code with the following values:
NOP=1 calculate first moment
NOP=2 calculate second moment
NOP=3 calculate third moment
NOP=4 calculate fourth moment
NOP=5 calculate all four moments

ANS(4). - Resultant vector containing the moments calculated.

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.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - lower bound greater than upper bound or number of intervals less than or equal to zero.

ERROR=2 - incorrect number of intervals for the specified bounds.

Method:

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

Mathematical Background:

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

where:

$$N = (UBO_3 - UBO_1)/UBO_2$$

UBO_1 = given lower bound

UBO_2 = given class interval

UBO_3 = given upper bound

and the total frequency

$$T = \sum_{i=1}^{N} F_i$$

where F_i = frequency in the i-th interval. If we set

$$X_i = UBO_1 - 0.5UBO_2 + i\ UBO_2 \qquad i=1,\ldots,n$$

then the first moment (mean)

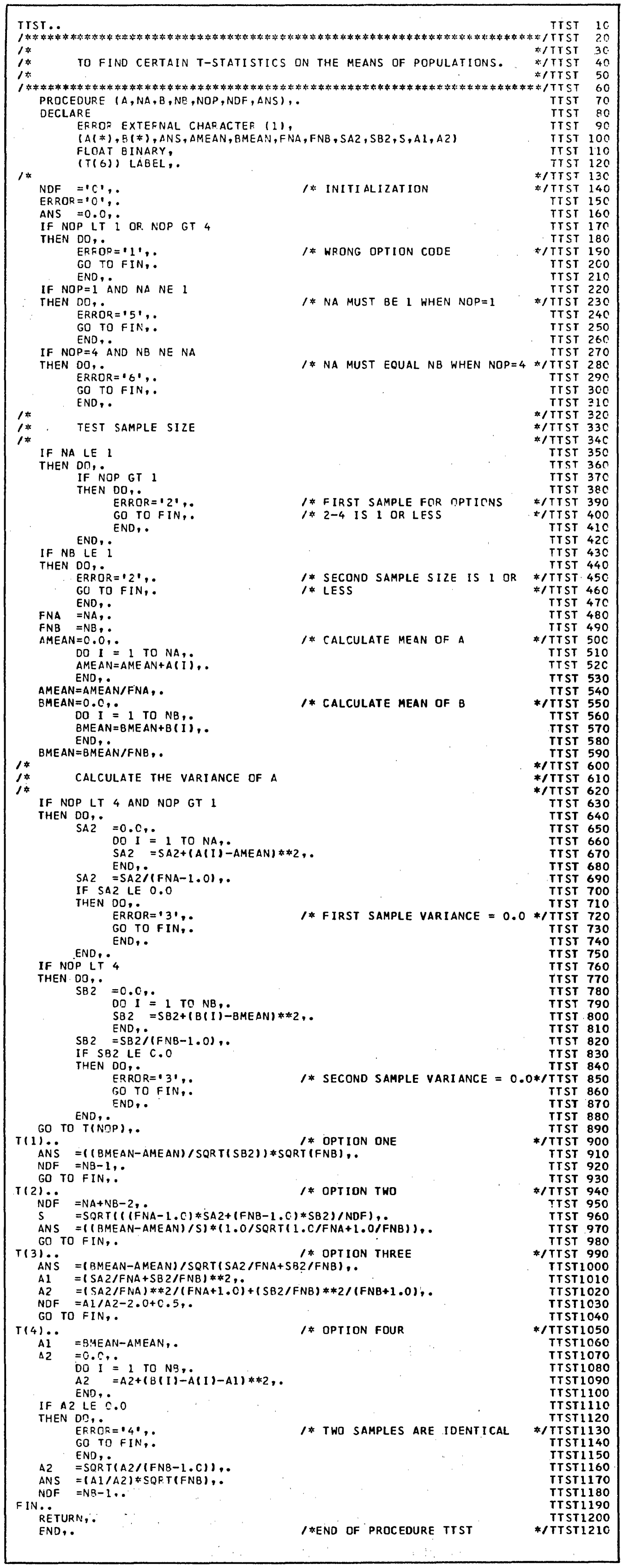
$$ANS_1 = 1/T \sum_{i=1}^{N} F_i X_i$$

and the j^th moment (j=2, 3, 4) is

$$ANS_j = 1/T \sum_{i=1}^{N} F_i (X_i - ANS_1)^j$$

These moments are biased and not corrected for grouping.

Subroutine TTST



```
TTST..
/*********************************************************************/
/*                                                                   */
/*     TO FIND CERTAIN T-STATISTICS ON THE MEANS OF POPULATIONS.     */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,NA,B,NB,NOP,NDF,ANS),.
   DECLARE
         ERROR EXTERNAL CHARACTER (1),
         (A(*),B(*),ANS,AMEAN,BMEAN,FNA,FNB,SA2,SB2,S,A1,A2)
         FLOAT BINARY,
         (T(6)) LABEL,.
/*                                                                   */
   NDF  ='0',.                    /* INITIALIZATION                  */
   ERROR='0',.
   ANS  =0.0,.
   IF NOP LT 1 OR NOP GT 4
   THEN DO,.
         ERROR='1',.              /* WRONG OPTION CODE               */
         GO TO FIN,.
         END,.
   IF NOP=1 AND NA NE 1
   THEN DO,.                      /* NA MUST BE 1 WHEN NOP=1         */
         ERROR='5',.
         GO TO FIN,.
         END,.
   IF NOP=4 AND NB NE NA
   THEN DO,.                      /* NA MUST EQUAL NB WHEN NOP=4     */
         ERROR='6',.
         GO TO FIN,.
         END,.
/*                                                                   */
/*       TEST SAMPLE SIZE                                            */
/*                                                                   */
   IF NA LE 1
   THEN DO,.
         IF NOP GT 1
         THEN DO,.
               ERROR='2',.        /* FIRST SAMPLE FOR OPTIONS        */
               GO TO FIN,.        /* 2-4 IS 1 OR LESS                */
               END,.
         END,.
   IF NB LE 1
   THEN DO,.
         ERROR='2',.              /* SECOND SAMPLE SIZE IS 1 OR      */
         GO TO FIN,.              /* LESS                            */
         END,.
   FNA  =NA,.
   FNB  =NB,.
   AMEAN=0.0,.                    /* CALCULATE MEAN OF A             */
         DO I = 1 TO NA,.
         AMEAN=AMEAN+A(I),.
         END,.
   AMEAN=AMEAN/FNA,.
   BMEAN=0.0,.                    /* CALCULATE MEAN OF B             */
         DO I = 1 TO NB,.
         BMEAN=BMEAN+B(I),.
         END,.
   BMEAN=BMEAN/FNB,.
/*                                                                   */
/*       CALCULATE THE VARIANCE OF A                                 */
/*                                                                   */
   IF NOP LT 4 AND NOP GT 1
   THEN DO,.
         SA2  =0.0,.
               DO I = 1 TO NA,.
               SA2  =SA2+(A(I)-AMEAN)**2,.
               END,.
         SA2  =SA2/(FNA-1.0),.
         IF SA2 LE 0.0
         THEN DO,.
               ERROR='3',.        /* FIRST SAMPLE VARIANCE = 0.0     */
               GO TO FIN,.
               END,.
         END,.
   IF NOP LT 4
   THEN DO,.
         SB2  =0.0,.
               DO I = 1 TO NB,.
               SB2  =SB2+(B(I)-BMEAN)**2,.
               END,.
         SB2  =SB2/(FNB-1.0),.
         IF SB2 LE 0.0
         THEN DO,.
               ERROR='3',.        /* SECOND SAMPLE VARIANCE = 0.0*/
               GO TO FIN,.
               END,.
         END,.
   GO TO T(NOP),.
T(1)..                            /* OPTION ONE                      */
   ANS  =((BMEAN-AMEAN)/SQRT(SB2))*SQRT(FNB),.
   NDF  =NB-1,.
   GO TO FIN,.
T(2)..                            /* OPTION TWO                      */
   NDF  =NA+NB-2,.
   S    =SQRT(((FNA-1.0)*SA2+(FNB-1.0)*SB2)/NDF),.
   ANS  =((BMEAN-AMEAN)/S)*(1.0/SQRT(1.0/FNA+1.0/FNB)),.
   GO TO FIN,.
T(3)..                            /* OPTION THREE                    */
   ANS  =(BMEAN-AMEAN)/SQRT(SA2/FNA+SB2/FNB),.
   A1   =(SA2/FNA+SB2/FNB)**2,.
   A2   =(SA2/FNA)**2/(FNA+1.0)+(SB2/FNB)**2/(FNB+1.0),.
   NDF  =A1/A2-2.0+0.5,.
   GO TO FIN,.
T(4)..                            /* OPTION FOUR                     */
   A1   =BMEAN-AMEAN,.
   A2   =0.0,.
         DO I = 1 TO NB,.
         A2   =A2+(B(I)-A(I)-A1)**2,.
         END,.
   IF A2 LE 0.0
   THEN DO,.
         ERROR='4',.              /* TWO SAMPLES ARE IDENTICAL       */
         GO TO FIN,.
         END,.
   A2   =SQRT(A2/(FNB-1.0)),.
   ANS  =(A1/A2)*SQRT(FNB),.
   NDF  =NB-1,.
FIN..
   RETURN,.
   END,.                          /*END OF PROCEDURE TTST            */
```

Purpose:

TTST calculates certain t-statistics on the means of populations.

Usage:

CALL TTST (A, NA, B, NB, NOP, NDF, ANS);

A(NA) - BINARY FLOAT
Given vector containing data.
NA - BINARY FIXED
Given number of observations in A.
B(NB) - BINARY FLOAT
Given vector containing data.
NB - BINARY FIXED
Given number of observations in B.
NOP - BINARY FIXED
Given options for various hypotheses:
NOP=1 - That population mean of B = given value of 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 the variance of B is not equal to the variance of A.
NOP=4 - That population mean of A = population mean of B, given no information about variance of A and B (set NA = NB).
NDF - BINARY FIXED
Resultant variable containing degrees of freedom associated with t-statistic calculated.
ANS - BINARY FLOAT
Resultant variable containing t-statistic.

Remarks:

NA and NB must be greater than one, 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.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - invalid option code.
ERROR=2 - sample size of one of the variables is less than or equal to 1 (except variable A in option 1).
ERROR=3 - variance of one of the variables is zero.
ERROR=4 - two samples identical.
ERROR=5 - NA must be 1 when NOP is 1.
ERROR=6 - NA must equal NB when NOP is 4.

Method:

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

Mathematical Background:

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

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

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

and the corresponding sample variances by:

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

The quantities μ and σ^2 stand respectively for population mean and variance in the following hypotheses.

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

Let $\bar{B}$ = estimate of μ_B and set NA = 1 (A is stored in location A_1).

The subroutine computes:

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

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

Hypothesis: $\mu_A = \mu_B; \left(\sigma_A^2 = \sigma_B^2\right)$ (option 2)

The subroutine computes:

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

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

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

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

The subroutine computes:

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

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

(degrees of freedom)

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

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

The subroutine computes:

$$ANS = \frac{\overline{D}}{SD} \cdot \sqrt{NB} \qquad \text{(t-statistics)} \qquad (10'$$

$$NDF = NB - 1 \qquad \text{(degrees of free'}$$

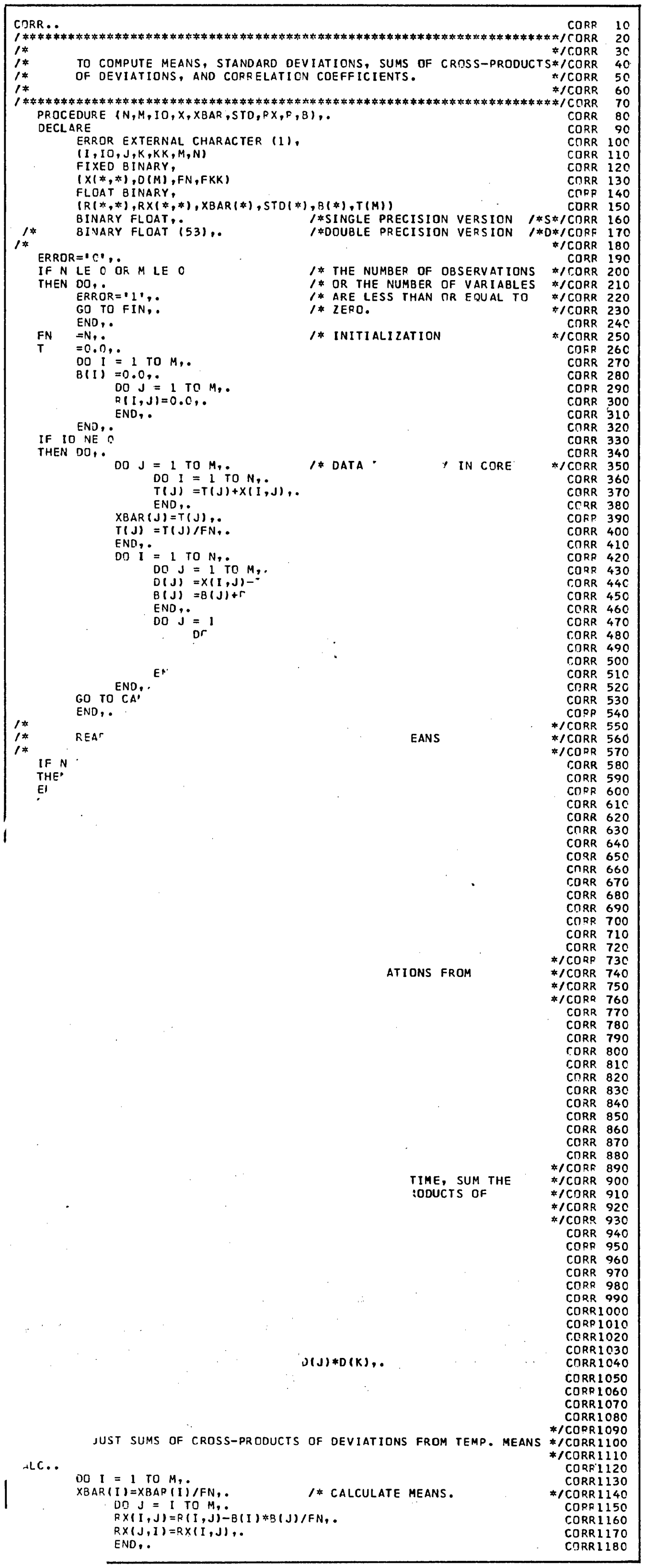
where $\overline{D} = \overline{B} - \overline{A}$

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

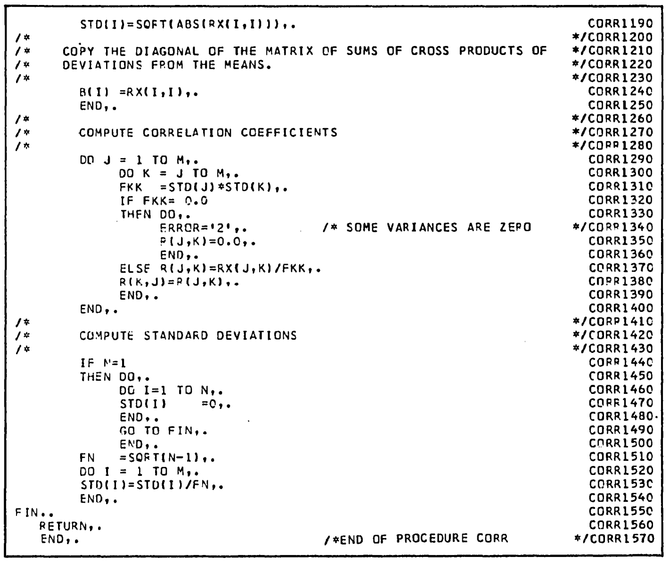
$NA = NB$

Correlation and Regression Analysis

- Subroutine CORR



```
CORR..
/*********************************************************************/
/*                                                                   */
/*      TO COMPUTE MEANS, STANDARD DEVIATIONS, SUMS OF CROSS-PRODUCTS*/
/*      OF DEVIATIONS, AND CORRELATION COEFFICIENTS.                 */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (N,M,IO,X,XBAR,STD,RX,R,B),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (I,IO,J,K,KK,M,N)
        FIXED BINARY,
        (X(*,*),D(M),FN,FKK)
        FLOAT BINARY,
        (R(*,*),RX(*,*),XBAR(*),STD(*),B(*),T(M))
        BINARY FLOAT,.                    /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.               /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                   */
   ERROR='C',.
   IF N LE 0 OR M LE 0                /* THE NUMBER OF OBSERVATIONS  */
   THEN DO,.                          /* OR THE NUMBER OF VARIABLES  */
        ERROR='1',.                   /* ARE LESS THAN OR EQUAL TO   */
        GO TO FIN,.                   /* ZERO.                       */
        END,.
   FN   =N,.                          /* INITIALIZATION              */
   T    =0.0,.
        DO I = 1 TO M,.
        B(I) =0.0,.
             DO J = 1 TO M,.
             R(I,J)=0.0,.
             END,.
        END,.
   IF IO NE 0
   THEN DO,.
             DO J = 1 TO M,.          /* DATA '         ' IN CORE    */
                  DO I = 1 TO N,.
                  T(J) =T(J)+X(I,J),.
                  END,.
             XBAR(J)=T(J),.
             T(J) =T(J)/FN,.
             END,.
             DO I = 1 TO N,.
                  DO J = 1 TO M,.
                  D(J) =X(I,J)-'
                  B(J) =B(J)+'
                  END,.
                  DO J = 1
                       D'



                  E'
             END,.
        GO TO CA'
        END,.
/*                                                                   */
/*      REA'                                          EANS           */
/*                                                                   */
   IF N '
   THE'
   E'










                                                                     */
                                                ATIONS FROM          */
                                                                     */
                                                                     */









                                                                     */
                                                   TIME, SUM THE     */
                                                   'ODUCTS OF        */
                                                                     */
                                                                     */









                                      D(J)*D(K),.





                                                                     */
      JUST SUMS OF CROSS-PRODUCTS OF DEVIATIONS FROM TEMP. MEANS */
                                                                     */
 'LC..
        DO I = 1 TO M,.
        XBAR(I)=XBAR(I)/FN,.          /* CALCULATE MEANS.            */
             DO J = I TO M,.
             RX(I,J)=R(I,J)-B(I)*B(J)/FN,.
             RX(J,I)=RX(I,J),.
             END,.
```



```
        STD(I)=SOFT(ABS(RX(I,I))),.                                      CORR1190
/*                                                                     */CORR1200
/*    COPY THE DIAGONAL OF THE MATRIX OF SUMS OF CROSS PRODUCTS OF     */CORR1210
/*    DEVIATIONS FROM THE MEANS.                                       */CORR1220
/*                                                                     */CORR1230
        B(I) =RX(I,I),.                                                  CORR1240
        END,.                                                            CORR1250
/*                                                                     */CORR1260
/*      COMPUTE CORRELATION COEFFICIENTS                               */CORR1270
/*                                                                     */CORP1280
        DO J = 1 TO M,.                                                  CORR1290
             DO K = J TO M,.                                             CORR1300
             FKK  =STD(J)*STD(K),.                                       CORR1310
             IF FKK= 0.0                                                 CORR1320
             THFN DO,.                                                   CORR1330
                  ERROR='2',.         /* SOME VARIANCES ARE ZEPO       */CORP1340
                  P(J,K)=0.0,.                                           CORR1350
                  END,.                                                  CORR1360
             ELSE R(J,K)=RX(J,K)/FKK,.                                   CORR1370
             R(K,J)=P(J,K),.                                             COPR1380
             END,.                                                       CORR1390
        END,.                                                            CORR1400
/*                                                                     */CORP1410
/*      COMPUTE STANDARD DEVIATIONS                                    */CORR1420
/*                                                                     */CORR1430
        IF N=1                                                           CORP1440
        THEN DO,.                                                        CORR1450
             DO I=1 TO N,.                                               CORR1460
             STD(I)    =0,.                                              CORR1470
             END,.                                                       CORR1480
             GO TO FIN,.                                                 CORR1490
             END,.                                                       CORR1500
        FN   =SQRT(N-1),.                                                CORR1510
        DO I = 1 TO M,.                                                  CORR1520
        STD(I)=STD(I)/FN,.                                               CORR1530
        END,.                                                            CORR1540
FIN..                                                                    CORR1550
   RETURN,.                                                              CORR1560
   END,.                              /*END OF PROCEDURE CORR          */CORR1570
```

Purpose:

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

Usage:

CALL CORR (N, M, IO, X, XBAR, STD, RX, R, B);

Description of parameters:

N - BINARY FIXED
Given number of observations.

M - BINARY FIXED
Given number of variables for each observation.

IO - BINARY FIXED
Given option code for input data.

X(N, M) - BINARY FLOAT
IO=0 If data are to be read in from input device in the special procedure named DAT2 (see "Remarks").
IO≠0 If all data are already in core.
If IO=0, X is not used.
If IO≠0, X is the input matrix containing data already in core.

XBAR(M) - BINARY FLOAT [(53)]
Resultant vector of length M containing means.

STD - BINARY FLOAT [(53)]
Resultant vector of length M containing standard deviations.

RX(M, M) - BINARY FLOAT [(53)]
Resultant matrix (M by M) containing sums of cross-products of deviations from means.

R(M, M) - BINARY FLOAT [(53)]
Resultant matrix (M by M) containing correlation coefficients.

B(M). - BINARY FLOAT [(53)]
Resultant vector of length M containing the diagonal of the matrix of sums of cross-products of deviations from means.

Remarks:

Subroutines and function subroutines required:

DAT2(M, D). This subroutine may be provided by the user, but a suitable subroutine is used in several of the sample programs (for example, see REGR). Note that in using this procedure, the parameters NCARD and NV must be declared external and the proper values must be assigned to them.

1. If IO=0, this subroutine provides an observation in vector D from an input device.

2. If IO≠0, this procedure is not used by CORR but must be in the job deck. If the user has neither supplied a subroutine nor used the subroutine DAT2 from the Scientific Subroutine Package, the following is suggested:

```
DAT2..
   PROCEDURE,.
   RETURN,.
   END,.
```

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations less than or equal to zero.
ERROR=2 - at least one variance is zero.

Method:

Product-moment correlation coefficients are computed.

Mathematical Background:

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

The following equations are used to calculate these statistics:

Sums of cross-products of deviations:

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

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

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

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

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

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

Correlation coefficients:

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

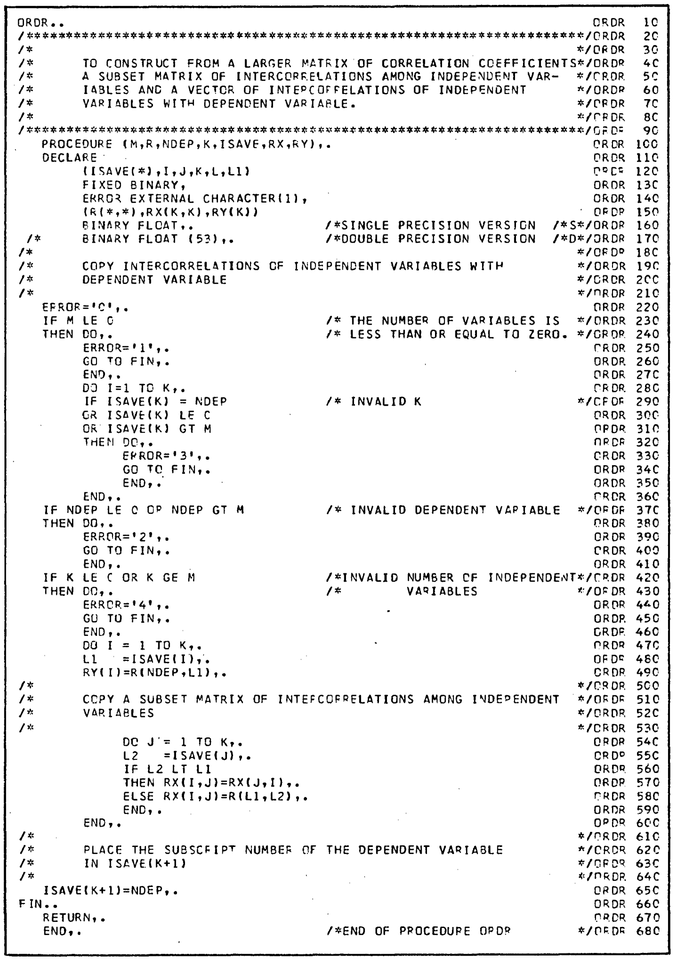
where j = 1, 2, ..., m; k = 1, 2, ..., m

Standard deviations:

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

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

- Subroutine ORDR



```
ORDR..
/**********************************************************************/
/*                                                                    */
/*      TO CONSTRUCT FROM A LARGER MATRIX OF CORRELATION COEFFICIENTS*/
/*      A SUBSET MATRIX OF INTERCORRELATIONS AMONG INDEPENDENT VAR-   */
/*      IABLES AND A VECTOR OF INTERCORRELATIONS OF INDEPENDENT       */
/*      VARIABLES WITH DEPENDENT VARIABLE.                            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,R,NDEP,K,ISAVE,RX,RY),.
   DECLARE
        (ISAVE(*),I,J,K,L,L1)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (R(*,*),RX(K,K),RY(K))
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
/*      COPY INTERCORRELATIONS OF INDEPENDENT VARIABLES WITH          */
/*      DEPENDENT VARIABLE                                            */
/*                                                                    */
   ERROR='0',.
   IF M LE 0                           /* THE NUMBER OF VARIABLES IS  */
   THEN DO,.                           /* LESS THAN OR EQUAL TO ZERO. */
        ERROR='1',.
        GO TO FIN,.
        END,.
        DO I=1 TO K,.
        IF ISAVE(K) = NDEP             /* INVALID K                   */
        OR ISAVE(K) LE 0
        OR ISAVE(K) GT M
        THEN DO,.
             ERROR='3',.
             GO TO FIN,.
             END,.
        END,.
   IF NDEP LE 0 OR NDEP GT M           /* INVALID DEPENDENT VARIABLE  */
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
   IF K LE 0 OR K GE M                 /*INVALID NUMBER OF INDEPENDENT*/
   THEN DO,.                           /*         VARIABLES           */
        ERROR='4',.
        GO TO FIN,.
        END,.
        DO I = 1 TO K,.
        L1   =ISAVE(I),.
        RY(I)=R(NDEP,L1),.
/*                                                                    */
/*      COPY A SUBSET MATRIX OF INTERCORRELATIONS AMONG INDEPENDENT   */
/*      VARIABLES                                                     */
/*                                                                    */
             DO J = 1 TO K,.
             L2   =ISAVE(J),.
             IF L2 LT L1
             THEN RX(I,J)=RX(J,I),.
             ELSE RX(I,J)=R(L1,L2),.
             END,.
        END,.
/*                                                                    */
/*      PLACE THE SUBSCRIPT NUMBER OF THE DEPENDENT VARIABLE          */
/*      IN ISAVE(K+1)                                                 */
/*                                                                    */
   ISAVE(K+1)=NDEP,.
FIN..
   RETURN,.
   END,.                               /*END OF PROCEDURE ORDR        */
```

Purpose:

ORDR is used to choose a dependent variable and a set of independent variables from a matrix of correlation coefficients, and form a submatrix of correlation coefficients to be used in performing a multiple linear regression analysis.

Usage:

CALL ORDR (M, R, NDEP, K, ISAVE, RX, RY);

Description of parameters:

M - BINARY FIXED
Given number of variables.

R(M, M) - BINARY FLOAT [(53)]
Given matrix containing correlation coefficients.

NDEP - BINARY FIXED
Given subscript number of the dependent variable.

K - BINARY FIXED
Given number of independent variables to be included in the forthcoming regression.

ISAVE - (K+1) BINARY FIXED
Given vector containing, in ascending order, the subscript numbers of K independent variables to be included in the forthcoming regression. Upon returning to the calling program, this vector contains, in addition, the subscript number of the dependent variable in K+1 position.

RX(K,K) - BINARY FLOAT [(53)]
Resultant matrix containing intercorrelations among independent variables to be used in forthcoming regression.

RY(K) - BINARY FLOAT [(53)]
Resultant vector containing intercorrelations of independent variables with dependent variables.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of variables less than or equal to zero.

ERROR=2 - invalid dependent variable subscript.

ERROR=3 - invalid independent variable subscript. If this condition exists, RX and RY will contain invalid values.

ERROR=4 - invalid number of independent variables.

Method:

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

- <u>Subroutine</u> MLTR

```
MLTR..
/***********************************************************************/
/*                                                                     */
/*      TO PERFORM A MULTIPLE LINEAR REGRESSION ANALYSIS FOR A         */
/*      DEPENDENT VARIABLE AND A SET OF INDEPENDENT VARIABLES.         */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (N,K,XBAR,STD,D,RX,RY,ISAVE,B,SB,T,BETA,ANS),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (I,IO,J,M,MM,MP,MQ,N,N1,ISAVE(*))
        FIXED BINARY,
        (XBAR(*),STD(*),D(*),RX(*,*),RY(*),B(*),SB(*),T(*),BETA(*),
        ANS(10),RM,BO,SSAR,SSDR,FK,FNN,SY,SSARM,SSDRM,F)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                     */
   ERROR='0',.
   IF K LE 0 OR N LE K                 /* THE NUMBER OF VARIABLES IS   */
   THEN DO,.                           /* LESS THAN OR EQUAL TO ZERO   */
        ERROR='1',.                    /* OR THE NO. OF OBSERVATIONS   */
        GO TO S10,.                    /* IS LESS THAN OR EQUAL TO THE*/
        END,.                          /* THE NUMBER OF VARIABLES      */
   MM   =K+1,.
   FK   =K,.
        DO J = 1 TO K,.
        BETA(J)=0.0,.
        B(J) =0.0,.
             DO I = 1 TO K,.
             BETA(J)=BETA(J)+RY(I)*RX(I,J),.
             END,.
        END,.
   RM   =0.0,.
   BO   =0.0,.
   L1   =ISAVE(MM),.
/*                                                                     */
/*      COEFFICIENT OF DETERMINATION                                   */
/*                                                                     */
        DO I = 1 TO K,.
        RM   =RM+BETA(I)*RY(I),.
/*                                                                     */
/*      TEST ACCURACY OF THE RESULT                                    */
/*                                                                     */
        IF RM LT 0 OR RM GT 1
        THEN DO,.
             ERROR='2',.               /* INVALID MULTIPLE R           */
             GO TO S10,.
             END,.
        L    =ISAVE(I),.               /* REGRESSION COEFFICIENT       */
        B(I) =BETA(I)*(STD(L1)/STD(L)),.
        BO   =BO+B(I)*XBAR(L),.        /* INTERCEPT                    */
        END,.
   BO   =XBAR(L1)-BO,.
/*                                                                     */
/*      SUM OF SQUARES ATTRIBUTED TO REGRESSION                        */
/*                                                                     */
   SSAR =RM*D(L1),.
   IF SSAR GT D(L1)                    /* TEST SUM OF SQUARES REDUCED */
   THEN DO,.
        ERROR='3',.                    /* REDUCED SUM OF SQUARES       */
        GO TO S10,.                    /* GREATER THAN THE TOTAL       */
        END,.                          /* SUM OF SQUARES               */
   RM   =SQRT(ABS(RM)),.               /* MULTIPLE CORRELATION COEFF. */
/*                                                                     */
/*      SUM OF SQUARES OF DEVIATIONS FROM REGRESSION                   */
/*                                                                     */
   SSDR =D(L1)-SSAR,.
   FNN  =N-K-1,.                       /* DEGREES OF FREEDOM           */
   IF FNN LE 0.0
   THEN DO,.
        ERROR='1',.                    /* SAMPLE SIZE TOO SMALL        */
        GO TO S10,.
        END,.
   SY   =SSDR/FNN,.                    /* VARIANCE OF ESTIMATE         */
/*                                                                     */
/*      STANDARD DEVIATIONS OF REGRESSION COEFFICIENTS                 */
/*                                                                     */
        DO J = 1 TO K,.
        L    =ISAVE(J),.
        SB(J)=SQRT(ABS((RX(J,J)/D(L))*SY)),.
        T(J) =B(J)/SB(J),.             /* COMPUTE T-VALUES             */
        END,.
   SY   =SQRT(ABS(SY)),.               /* STANDARD ERROR OF ESTIMATE   */
   SSARM=SSAR/FK,.                     /* F-VALUE                      */
   SSDRM=SSDR/FNN,.
   F    =SSARM/SSDRM,.
   ANS(1)=BO,.
   ANS(2)=RM,.
   ANS(3)=SY,.
   ANS(4)=SSAR,.
   ANS(5)=FK,.
   ANS(6)=SSARM,.
   ANS(7)=SSDR,.
   ANS(8)=FNN,.
   ANS(9)=SSDRM,.
   ANS(10)=F,.
S10..
   RETURN,.
   END,.                               /*END OF PROCEDURE MLTR         */
```

Purpose:

MLTR performs a multiple linear regression analysis for a dependent variable and a set of independent variables.

Usage:

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

Description of parameters:

N - BINARY FIXED
Given number of observations (N must be greater than K).

K - BINARY FIXED
Given number of independent variables in this regression.

XBAR(M) - BINARY FLOAT [(53)]
Given vector containing means of all variables. M is the number of variables in an observation.

STD(M) - BINARY FLOAT [(53)]
Given vector containing standard deviations of all variables.

D(M) - BINARY FLOAT [(53)]
Given vector containing the diagonal of the matrix of sums of cross-products of deviations from means for all variables.

RX(K,K) - BINARY FLOAT [(53)]
Given matrix containing the inverse of intercorrelations among independent variables.

RY(K) - BINARY FLOAT [(53)]
Given vector containing intercorrelations of independent variables with dependent variable.

ISAVE - (K+1) BINARY FIXED
Given vector containing subscripts of independent variables in ascending order. The subscript of the dependent variable is stored in the last, K+1, position.

B(K) - BINARY FLOAT [(53)]
Resultant vector containing regression coefficients.

SB(K) - BINARY FLOAT [(53)]
Resultant vector containing standard deviations of regression coefficients.

T(K) - BINARY FLOAT [(53)]
Resultant vector containing T values.

BETA(K) - BINARY FLOAT [(53)]
Resultant vector containing beta coefficients.

ANS(10) - BINARY FLOAT [(53)]
Resultant vector 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:

If there are no errors in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of independent variables K less than or equal to zero or the number of observations N is less than or equal to K.

ERROR=2 - coefficient of determination (RM) less than zero or greater than one.

ERROR=3 - reduced sum of squares (SSAR) greater than the total sum of squares.

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.

Mathematical Background:

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:

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

where:

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

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

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

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

Then the regression coefficients are calculated as follows:

$$b_j = \beta_j \cdot \frac{s_y}{s_j} \tag{2}$$

where:

s_y = standard deviation of dependent variable

s_j = standard deviation of j-th independent variable

j = 1, 2, ..., k

s_y and s_j are input to this subroutine.

The intercept is found by the following equation:

$$b_0 = \overline{Y} - \sum_{j=1}^{k} b_j \cdot \overline{X}_j \tag{3}$$

where:

$\overline{Y}$ = mean of dependent variable

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

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

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

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

and taking the square root of R^2:

$$R = \sqrt{R^2} \tag{5}$$

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

$$SSAR = R^2 \cdot D_{yy} \tag{6}$$

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:

$$SSDR = D_{yy} - SSAR \tag{7}$$

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

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

Certain other statistics are calculated as follows:

Variance and standard error of estimate:

$$S^2_{y \cdot 12 \ldots k} = \frac{SSDR}{n-k-1} \tag{9}$$

where n = number of observations

$$S_{y \cdot 12 \ldots k} = \sqrt{S^2_{y \cdot 12 \ldots k}} \tag{10}$$

Standard deviations of regression coefficients:

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

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

j = 1, 2, ..., k

Computed t:

$$t_j = \frac{b_j}{S_{b_j}} \tag{12}$$

j = 1, 2, ..., k

• Subroutine STRG

```
STRG..
/**********************************************************************/
/*                                                                    */
/*      TO PERFORM A STEP-WISE MULTIPLE REGRESSION ANALYSIS FOR A     */
/*      DEPENDENT VARIABLE AND A SET OF INDEPENDENT VARIABLES.        */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,N,D,XBAR,IDX,PCT,NSTEP,ANS,L,B,STD),.
   DECLARE
        (I,ID,IJ,IK,J,K,KK,M,MK,MX,MY,N,NEW,NFO,NZ,NSTEP(*),IDX(*),
        L(*),LL(M))
        FIXED BINARY,
        (D(*,*),XBAR(*),ANS(*),B(*),STD(*),T(M),S(M),BETA(M),RE)
        BINARY FLOAT,                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),             /*DOUBLE PRECISION VERSION  /*D*/
        (PCT,ONM,RD)
        FLOAT BINARY,
        (ERROR,NSTOP) EXTERNAL CHARACTER (1),.
/*                                                                    */
   ERROR='0',.                         /* INITIALIZATION              */
   IF M LE 1 OR N LE M+1               /* THE NUMBER OF VARIABLES M IS*/
   THEN DO,.                           /* NOT GREATER THAN 1 OR THE   */
        ERROR='1',.                    /* NUMBER OF OBSERVATIONS N IS */
        GO TO S150,.                   /* NOT GREATER THAN M+1        */
        END,.
   IF PCT GE 1.0
   THEN DO,.
        ERROR='4',.                    /* SPECIFIED CONSTANT IS       */
        GO TO S150,.                   /* GREATER THAN OR = 1.0       */
        END,.
   ONM  =N-1,.
   NFO  =0,.
   NSTEP(3)=0,.
   ANS(3)=0,.
   ANS(4)=0,.
   NSTOP='0',.
/*                                                                    */
/*      FIND DEPENDENT VARIABLE, NUMBER OF VARIABLES TO BE FORCED TO */
/*      ENTER IN THE REGRESSION, AND THE NUMBER OF VARIABLES TO BE    */
/*      DELETED                                                       */
/*                                                                    */
        DO I = 1 TO M,.
        LL(I)=1,.
        IF IDX(I) LE 0
        THEN GO TO S10,.
        IF IDX(I) LT 2
        THEN DO,.
             NFO  =NFO+1,.
             IDX(NFO)=I,.
             GO TO S10,.
             END,.
        ELSE IF IDX(I)= 2
        THEN DO,.
             NSTEP(3)=NSTEP(3)+1,.
             LL(I)=-1,.
             GO TO S10,.
             END,.
        MY   =I,.
        NSTEP(1)=MY,.
        ANS(5)=D(MY,MY),.
S10..
        END,.
   NSTEP(2)=NFO,.
/*                                                                    */
/*      FIND THE MAXIMUM NUMBER OF STEPS                              */
/*                                                                    */
   MX   =M-NSTEP(3)-1,.
/*                                                                    */
/*      START SELECTION OF VARIABLES                                  */
/*                                                                    */
        DO NZ = 1 TO MX,.
        IF N-NZ-1 LE 0
        THEN DO,.
             ERROR='3',.               /* DEGREES OF FREEDOM IS 0     */
             GO TO S150,.
             END,.
        RD   =0,.
        IF NZ GT NFO
/*                                                                    */
/*      SELECT NEXT VARIABLE TO ENTER AMONG FORCED VARIABLES          */
/*                                                                    */
        THEN GO TO S20,.
             DO I = 1 TO NFO,.
             K    =IDX(I),.
             IF LL(K) GT 0
             THEN DO,.
                  RE   =D(K,MY)**2/D(K,K),.
                  IF RD LT RE
                  THEN DO,.
                       RD   =RE,.
                       NEW  =K,.
                       END,.
                  END,.
             END,.
        GO TO S25,.
/*                                                                    */
/*      SELECT NEXT VARIABLE TO ENTER AMONG NON-FORCED VARIABLES      */
/*                                                                    */
S20..
             DO I = 1 TO M,.
             IF I NE MY
             THEN DO,.
                  IF LL(I) GT 0
                  THEN DO,.
                       RE   =D(I,MY)**2/D(I,I),.
                       IF RD LT RE
                       THEN DO,.
                            RD   =RE,.
                            NEW  =I,.
                            END,.
                       END,.
                  END,.
             END,.
S25..
        IF RD LE 0 OR  ANS(5) LE ANS(3)+RD
        THEN DO,.
             ERROR='2',.               /* NEGATIVE SUM OF SQUARES     */
             GO TO S150,.
             END,.
        RE   =RD/ANS(5),.
/*                                                                    */
/*      TEST WHETHER THE PROPORTION OF THE SUM OF SQUARES REDUCED BY */
/*      THE LAST VARIABLE ENTERED IS GREATER THAN OR EQUAL TO THE     */
/*      SPECIFIED PROPORTION                                          */
/*                                                                    */
        IF RE LT PCT
        THEN GO TO S150,.
        LL(NEW)=0,.                    /* IT IS GREATER THAN OR EQUAL */
        L(NZ)=NEW,.
        ANS(1)=RD,.
        ANS(2)=RE,.
        ANS(3)=ANS(3)+RD,.
        ANS(4)=ANS(4)+RE,.
        NSTEP(4)=NZ,.
        NSTEP(5)=NEW,.
/*                                                                    */
/*      COMPUTE MULTIPLE CORRELATION, F-VALUE FOR ANALYSIS OF         */
/*      VARIANCE, AND STANDARD ERROR OF ESTIMATE                      */
/*                                                                    */
        ANS(6)=SQRT(ANS(4)),.
        RD   =NZ,.
        RE   =ONM-RD,.
        RE   =(ANS(5)-ANS(3))/RE,.
        ANS(7)=(ANS(3)/RD)/RE,.
        ANS(8)=SQRT(RE),.
/*                                                                    */
/*      DIVIDE BY THE PIVOTAL ELEMENT                                 */
/*                                                                    */
        RD   =D(NEW,NEW),.
             DO J = 1 TO M,.
             IF LL(J) LT 0
             THEN GO TO S40,.
             ELSE IF LL(J) GT 0
             THEN GO TO S30,.
             IF J = NEW
             THEN DO,.
                  D(NEW,NEW)=1/RD,.
                  GO TO S40,.
                  END,.
             D(J,J)=D(J,J)+D(NEW,J)**2/RD,.
S30..
             D(NEW,J)=D(NEW,J)/RD,.
S40..
             END,.
/*                                                                    */
/*      COMPUTE REGRESSION COEFFICIENTS                               */
/*                                                                    */
        B(NZ)=D(NEW,MY),.
        IF NZ GT 1
        THEN DO,.
             ID   =NZ-1,.
                  DO J = 1 TO ID,.
                  IJ   =NZ-J,.
                  KK   =L(IJ),.
                  B(IJ)=D(KK,MY),.
                       DO K = 1 TO J,.
                       IK   =NZ-K+1,.
                       MK   =L(IK),.
                       B(IJ)=B(IJ)-D(KK,MK)*B(IK),.
                       END,.
                  END,.
             END,.
        ANS(9)=XBAR(MY),.              /* COMPUTE INTERCEPT           */
             DO I = 1 TO NZ,.
             KK   =L(I),.
             ANS(9)=ANS(9)-B(I)*XBAR(KK),.
             S(I) =ANS(8)*SQRT(D(KK,KK)),.
             T(I) =B(I)/S(I),.
             BETA(I)=B(I)*STD(KK)/STD(MY),.
             END,.
/*                                                                    */
/*      PERFORM A REDUCTION TO ELIMINATE THE LAST VARIABLE ENTERED    */
/*                                                                    */
             DO I = 1 TO M,.
             IF LL(I) GT 0
             THEN DO,.
                       DO J = 1 TO M,.
                       IF LL(J) GE 0
                       THEN DO,.
                            IF J NE NEW
                            THEN D(I,J)=D(I,J)-D(I,NEW)*D(NEW,J),.
                            END,.
                       END,.
                  D(I,NEW)=D(I,NEW)/(-RD),.
                  END,.
             END,.
/*                                                                    */
/*      ADJUST STANDARD ERROR OF THE ESTIMATE AND MULTIPLE            */
/*      CORRELATION COEFFICIENT                                       */
/*                                                                    */
        RD   =N-NSTEP(4),.
        RD   =ONM/RD,.
        ANS(10)=SQRT(1-(1-ANS(6)**2)*RD),.
        ANS(11)=ANS(8)*SQRT(RD),.
        CALL SOUT (NSTEP,ANS,L,B,S,T,BETA),.
/*                                                                    */
/*      TEST WHETHER THE STEP-WISE REGRESSION WAS TERMINATED          */
/*      IN PROCEDURE SOUT.                                            */
/*                                                                    */
        IF NSTOP GT '0'
        THEN GO TO S150,.
        END,.
S150..
   RETURN,.
   END,.                               /*END OF PROCEDURE STRG        */
```

Purpose:

STRG performs a stepwise multiple linear regression analysis for a dependent variable and a set of independent variables.

Usage:

CALL STRG (M, N, D, XBAR, IDX, PCT, NSTEP, ANS, L, B, STD);

Description of parameters:

M - BINARY FIXED
Given total number of variables in data matrix.

N - BINARY FIXED
Given number of observations.

D(M, M) - BINARY FLOAT [(53)]
Given matrix of sums of cross-products of deviations from mean. This matrix will be destroyed.

XBAR(M) - BINARY FLOAT [(53)]
Given vector containing the means.

IDX(M) - BINARY FIXED
Given vector containing the following codes:
0 - independent variable available for selection.
1 - independent variable to be forced into the regression equation.
2 - variable not to be considered in the regression equation.
3 - dependent variable.
This input vector is destroyed.

PCT - BINARY FLOAT
Given constant value indicating the proportion of the total variance to be explained by any independent variable. Those independent variables that fall below this proportion will not enter the regression equation. To ensure that all variables enter the regression equation, set PCT=0.0.

NSTEP(5) - BINARY FIXED
Resultant vector containing the following information:

NSTEP(1) - number of the dependent variable.
NSTEP(2) - number of variables forced into the regression equation.
NSTEP(3) - number of variables deleted from the regression equation.
NSTEP(4) - the number of the last step.
NSTEP(5) - the number of the last variable entered.

ANS(11) - BINARY FLOAT [(53)]
Resultant vector containing the following information for the last step:

ANS(1) - Sum of squares reduced by this step
ANS(2) - Proportion of total sum of squares reduced
ANS(3) - Cumulative sum of squares reduced, up to this step
ANS(4) - Cumulative proportion of total sum of squares reduced
ANS(5) - Sum of squares of the dependent variable
ANS(6) - Multiple correlation coefficients
ANS(7) - F ratio for sum of squares due to regression
ANS(8) - Standard error of the estimate (residual mean square)
ANS(9) - Intercept constant
ANS(10) - Multiple correlation coefficient adjusted for degrees of freedom

L(K) - BINARY FIXED
Resultant vector containing the independent variables entered in the regression.
K is the number of independent variables in the regression equation.

B(K) - BINARY FLOAT [(53)]
Resultant vector containing the partial regression coefficients corresponding to the variables in vector L.

STD(M) - BINARY FLOAT [(53)]
Given vector containing the standard deviations.

Remarks:

There must be one, and only one, dependent variable and at least one independent variable.

The number of data points must be greater than the number of independent variables plus one. Forced variables are entered into the regression equation before all other independent variables. Within the set of forced variables, the one to be chosen first will be the one that explains the greater amount of variance.

Instead of using, as a stopping criterion, a proportion of the total variance, some other criterion may be added to the output routine.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of variables M not greater than 1, or N not greater than M+1.

ERROR=2 - reduced sum of squares exceeds total sum of squares.
ERROR=3 - degrees of freedom is zero, for the variable that is currently active.
ERROR=4 - specified constant, PCT, is greater than or equal to one.

Subroutines and function subroutines required: SOUT, a special output routine that must be provided by the user. The routine prints out the results of the stepwise regression. An example of such a routine may be found in the sample program STEP.

Method:

The abbreviated Doolittle method is used to (1) select variables entering the regression and (2) compute regression coefficients. Refer to C. A. Bennet and N. L. Franklin, Statistical Analysis in Chemistry and the Chemical Industry, John Wiley and Sons, 1954, Appendix 6A.

Mathematical Background:

This subroutine performs a stepwise multiple regression analysis for a dependent variable and a set of independent variables. In each step of the regression i=1,2,...,q, where q is the number of independent variables, the abbreviated Doolittle method is used to calculate the following statistics:

The independent variable entering in the regression is selected, first, by computing the amount of reduction of sum of squares for each variable:

$$C_j = \frac{a_{jy}^2}{a_{jj}} \qquad (1)$$

where:

a_{jj} is initially an element in the sums of cross-products of deviations matrix which will be modified in successive steps.

j = 1,2,..., q are independent variables (j ≠ variables deleted and variables entered before the i-th step)

y = dependent variable

and, second, by finding the largest value of C_j.

Set $S_i = C_j$ to indicate the sum of squares that will be reduced in the i-th step.

The proportion of S_i to the total is obtained by:

$$P = \frac{S_i}{D} \qquad (2)$$

where:

$$D = \sum_{j=1}^{n} (y_j - \bar{y})^2$$

(n = number of observations)

If p is less than the constant specified by the user to limit independent variables, the analysis will be terminated without entering the last variable selected; otherwise, the following calculations are continued:

The cumulative sum of squares reduced is obtained by

$$S_{cum} = S_{cum} + S_i \qquad (3)$$

and the cumulative proportion reduced by

$$P_{cum} = P_{cum} + p \qquad (4)$$

The multiple correlation coefficient is computed by

$$R = \sqrt{P_{cum}} \qquad (5)$$

and adjusted for degrees of freedom by

$$R_c = \sqrt{1-(1-R^2)\ (n-1)\ /\ (n-k)}$$

where there are k independent variables in the regression.

The F value for analysis of variance is given by

$$F = \frac{S_{cum}\ /\ k}{(D-S_{cum})\ /\ (n-k-1)} \qquad (6)$$

The standard error of the estimated y is obtained by the use of the formula

$$s_{y.12...i} = \sqrt{\frac{D - S_{cum}}{n - k - 1}} \qquad (7)$$

and adjusted by

$$s_c = s\ \sqrt{(n-1)\ /\ (n-k)}$$

Then the following is computed:

$$a_{jj} = a_{jj} + \frac{a_{ij}^2}{a_{ii}} \qquad (8)$$

where:

i = variable entered in the i-th step

j = $v_1, v_2, \ldots, v_{i-1}$ are the variables entered in the regression before the i-th step, and

$$g_{ik} = \frac{a_{ik}}{a_{ii}} \tag{9}$$

where k = 1, 2, ..., m are variables including y (k ≠ variables deleted or the variable entered in the i-th step).

Regression coefficients are computed by

$$b_i = g_{iy}$$

$$b_{i-1} = g_{(i-1)y} - b_i \, g_{(i-1)\,i} \tag{10}$$

$$b_{i-2} = g_{(i-2)y} - b_i g_{(i-2)\ i} - b_{i-1} g_{(i-2)\ (i-1)}$$

etc.

and the value of the intercept as

$$b_0 = \bar{y} - \sum_{j=1}^{k} b_j \, \bar{x}_j \tag{11}$$

where k = number of independent variables in the regression.

Standardized regression coefficients, beta weights

$$B_j = b_j \cdot \frac{S_j}{S_y} \tag{12}$$

where S_j and S_y are standard deviations.

Standard errors of regression coefficients are given by

$$s_{b_j} = \sqrt{a_{jj}} \cdot s_{y.12\ldots i} \tag{13}$$

where j = $v_1, v_2, \ldots, v_i$ are variables in the regression and t-values as

$$t_j = \frac{b_j}{s_{b_j}} \tag{14}$$

Perform the reduction to eliminate the variable entered in i-th step:

$$a_{jk} = a_{jk} - a_{ji} g_{ik} \tag{15}$$

where:

i = variable entered in i-th step

j = 1, 2, ..., m (j ≠ variables deleted and variables in the regression)

k = 1, 2, ..., m (k ≠ variables deleted or the variable entered in i-th step)

$$a_{ji} = a_{ji} / -a_{ii} \tag{16}$$

$$a_{ii} = 1 / a_{ii} \tag{17}$$

Programming Considerations:

If the user provides the routine SOUT, the argument list must be consistent with the argument list of the call statement in subroutine STRG.

A description of the parameters follows:

NSTEP(5), ANS(11) - L(K), B(K) These parameters are the same as in STRG. When used in SOUT, however, they appear as input.

S(M) - BINARY FLOAT [(53)]
Given vector containing standard error of regression.

T(M) - BINARY FLOAT [(53)]
Given computed T value.

BETA(M) - BINARY FLOAT [(53)]
Given beta coefficient.

Subroutine CANC

```
CANC..
/**********************************************************************/
/*                                                                    */
/*      TO COMPUTE THE CANONICAL CORRELATIONS BETWEEN TWO SETS OF    */
/*      VARIABLES.                                                    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (N,MP,MQ,RR,ROOTS,WLAM,CANR,CHISQ,NDF,COEFR,COEFL),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (NDF(*),I,J,K,L,M,MP,MQ,N,N1,IERR)
        FIXED BINARY,
        (RR(*,*),ROOTS(*),WLAM(*),CANR(*),CHISQ(*),COEFR(*,*),
        COEFL(*,*),DET,BAT,CON)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
/*      CHECK WHETHER THE NUMBER OF LEFT-HAND VARIABLES IS EQUAL TO  */
/*      OR GREATER THAN THAT OF RIGHT-HAND                            */
/*                                                                    */
   ERROR='0',.
   IERR =0,.
   IF MP LE 0 OR MQ LE 0               /* THERE ARE NO RIGHT OR LEFT  */
   THEN DO,.                           /* HAND VARIABLES.             */
        ERROR='1',.
        GO TO FIN,.
        END,.
   IF MP LT MQ
   THEN DO,.
        M    =MP,.
        MP   =MQ,.
        MQ   =M,.
        ERROR='2',.
        END,.
COPY..
   BEGIN,.
   DECLARE
        (R(MP,MP),T(MP,MQ),A(MQ,MQ),X(MQ,MQ))
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT(53),.             /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
/*      PARTITION INTERCORRELATIONS AMONG LEFT HAND VARIABLES,BETWEEN*/
/*      LEFT AND RIGHT HAND VARIABLES, AND AMONG RIGHT HAND VARIABLES*/
/*                                                                    */
   M    =MP+MQ,.
   FM   =M+1,.
   FN   =N,.
   IF ERROR= '2'
   THEN DO,.
        IERR =1,.
        K    =0,.                      /* CHANGE LEFT AND RIGHT HAND  */
             DO I = MQ+1 TO M,.        /* VARIABLES                   */
             K    =K+1,.
             L    =0,.
                  DO J = MQ+1 TO M,.
                  L    =L+1,.
                                       /* RR 22 INTO R                */
             R(K,L)=RR(I,J),.
                  END,.
                                       /* RR 21 INTO COEFL            */
                  DO J = 1 TO MQ,.
                  COEFL(K,J)=RR(I,J),.
                  END,.
             END,.
             DO I = 1 TO MQ,.
                  DO J = 1 TO MQ,.
                                       /* RR 11 INTO COEFR            */
                 COEFR(I,J)=RR(I,J),.
                  END,.
             END,.
        END,.
   ELSE DO,.
             DO I = 1 TO M,.
                  DO J = 1 TO M,.
                  IF I LE MP AND J LE MP
                  THEN DO,.
                                       /* RR 11 INTO R                */
                       R(I,J)=RR(I,J),.
                       GO TO S10,.
                       END,.
                  IF I LE MP AND J GT MP
                  THEN DO,.
                       K    =J-MP,.
                                       /* RR 12 INTO COEFL            */
                       COEFL(I,K)=RR(I,J),.
                       GO TO S10,.
                       END,.
                  IF I GT MP AND J GT MP
                  THEN DO,.
                       L    =I-MP,.
                       K    =J-MP,.
                                       /* RR 22 INTO COEFR            */
                       COEFR(L,K)=RR(I,J),.
                       END,.
S10..
                  END,.
             END,.
/*                                                                    */
/*      SOLVE THE CANONICAL EQUATION                                  */
/*                                                                    */
        CON  =0,.
        CALL MINV (R,MP,DET,CON),.
        IF ERROR NE '0'
        THEN DO,.
             ERROR='3',.
             GO TO FIN,.
             END,.
/*                                                                    */
/*      CALCULATE T = INVERSE OF RR 11 * RR 12                        */
/*                                                                    */
             DO I = 1 TO MP,.
                  DO J = 1 TO MQ,.
                  T(I,J)=0.0,.
                       DO K = 1 TO MP,.
                       T(I,J)=T(I,J)+R(I,K)*COEFL(K,J),.
                       END,.
                  END,.
             END,.
/*                                                                    */
/*      CALCULATE  A = RR 21 * T                                      */
/*                                                                    */
             DO I = 1 TO MQ,.
                  DO J = 1 TO MQ,.
                  A(I,J)=0.0,.
                       DO K = 1 TO MP,.
                       A(I,J)=A(I,J)+COEFL(K,I)*T(K,J),.
                       END,.
                  END,.
             END,.
/*                                                                    */
/*      CALCULATE EIGENVALUES WITH ASSOCIATED EIGENVECTORS OF THE    */
/*      INVERSE OF R 22 * A                                           */
/*                                                                    */
        CALL MGDU (MQ,A,COEFR,ROOTS,X),.
        IF ERROR NE '0'
        THEN DO,.
             ERROR='4',.               /* ERROR CONDITION IN ROUTINE  */
             GO TO FIN,.               /* MSDU.                       */
             END,.
        IF IERR= '1'
        THEN ERROR='2',.
/*                                                                    */
/*      TEST WHETHER EIGENVALUES ARE GREATER THAN 0.0 BUT LESS THAN  */
/*      1.0                                                           */
/*                                                                    */
             DO I = 1 TO MQ,.
             IF ROOTS(I) LE 0.0 OR ROOTS(I) GE 1.0
             THEN DO,.
                  ERROR='5',.          /* CANONICAL CORRELATION CANNOT*/
                  GO TO FIN,.          /* BE COMPUTED                 */
                  END,.
             END,.
/*                                                                    */
/*      FOR EACH VALUE OF I = 1,2,...,MQ CALCULATE THE STATISTICS    */
/*      NOTED BELOW.                                                  */
/*                                                                    */
             DO I = 1 TO MQ,.
                                       /* CANONICAL CORRELATION       */
             CANR(I)=SQRT(ROOTS(I)),.
             WLAM(I)=1.0,.
                  DO J = I TO MQ,.
                  WLAM(I)=WLAM(I)*(1.0-ROOTS(J)),.
                  END,.
             BAT  =WLAM(I),.           /* CHI-SQUARE                  */
             CHISQ(I)=-(FN-0.5*FM)*LOG(BAT),.
/*                                                                    */
/*      CALCULATE DEGREES OF FREEDOM FOR CHI-SQUARE                   */
/*                                                                    */
             N1  =I-1,.
             NDF(I)=(MP-N1)*(MQ-N1),.
/*                                                                    */
/*      I-TH SET OF RIGHT HAND COEFICIENTS                            */
/*                                                                    */
                  DO J = 1 TO MQ,.
                  COEFR(J,I)=X(J,I),.
                  END,.
/*                                                                    */
/*      I-TH SET OF LEFT HAND COEFFICIENTS                            */
/*                                                                    */
                  DO J = 1 TO MP,.
                  DET  =0.0,.
                       DO K = 1 TO MQ,.
                       DET  =DET+T(J,K)*COEFR(K,I),.
                       END,.
                  COEFL(J,I)=DET/CANR(I),.
                  END,.
             END,.
        END,.
   END,.
FIN..
   RETURN,.
   END,.                               /*END OF PROCEDURE CANC        */
```

Purpose:

CANC computes the canonical correlations between two sets of variables. It is normally preceded by a call to procedure CORR.

Usage:

CALL CANC (N, MP, MQ, RR, ROOTS, WLAM, CANR, CHISQ, NDF, COEFR, COEFL);

Description of parameters:

N - BINARY FIXED
Given number of observations.

MP - BINARY FIXED
Given number of left hand variables.

MQ - BINARY FIXED
Given number of right hand variables.

RR(M, M) - BINARY FLOAT [(53)]
Given matrix (where M=MP+MQ), containing correlation coefficients.

ROOTS(MQ) - BINARY FLOAT [(53)]
Resultant vector containing eigenvalues computed in the subroutine MGDU.

WLAM(MQ) - BINARY FLOAT [(53)]
Resultant vector of length MQ containing lambda.

CANR(MQ) - BINARY FLOAT [(53)]
Resultant vector containing canonical correlations.

CHISQ(MQ) - BINARY FLOAT [(53)]
Resultant vector containing the values of chi-squares.

NDF - BINARY FIXED
Resultant variable containing the number of degrees of freedom.

COEFR - (MQ, MQ) BINARY FLOAT [(53)]
Resultant matrix containing MQ sets of right-hand coefficients columnwise.

COEFL - (MP, MQ) BINARY FLOAT [(53)]
Resultant matrix containing MQ sets of left-hand coefficients columnwise.

Remarks:

The number of left-hand variables (MP) should be greater than or equal to the number of right-hand variables (MQ). If the value of MP is less than the value of MQ, the input matrix is rearranged to satisfy the above conditions. The right-hand variables become left-hand variables and left-hand variables become right-hand variables. If this condition exists, the error code indicator, ERROR, is set to 2.

Also, if the variables are changed, the values of MP and MQ are interchanged.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - no right-hand or left-hand variable -- returned values are meaningless.
ERROR=2 - number of left-hand variables smaller than the number of right-hand variables.
ERROR=3 - correlation coefficient matrix ill-conditioned (determined by MINV).
ERROR=4 - error condition in routine MGDU, from MSDU.
ERROR=5 - Eigenvalues less than or equal to zero or greater than or equal to one.

Subroutines and function subroutines required:

MINV
MGDU (which, in turn, calls the subroutine MSDU)

Method:

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

Mathematical Background:

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

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

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

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

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

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

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

The equation:

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

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

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

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

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

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

Canonical correlation:

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

where λ_i = i-th eigenvalue

Chi-square:

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

where n = number of observations

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

Degrees of freedom for χ^2:

$$DF = \left[p-(i-1)\right]\ \left[q-(i-1)\right] \tag{7}$$

i-th set of right-hand coefficients:

$$b_k = v_{ki} \tag{8}$$

where v_{ki} = eigenvector associated with λ_i

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

i-th set of left-hand coefficients:

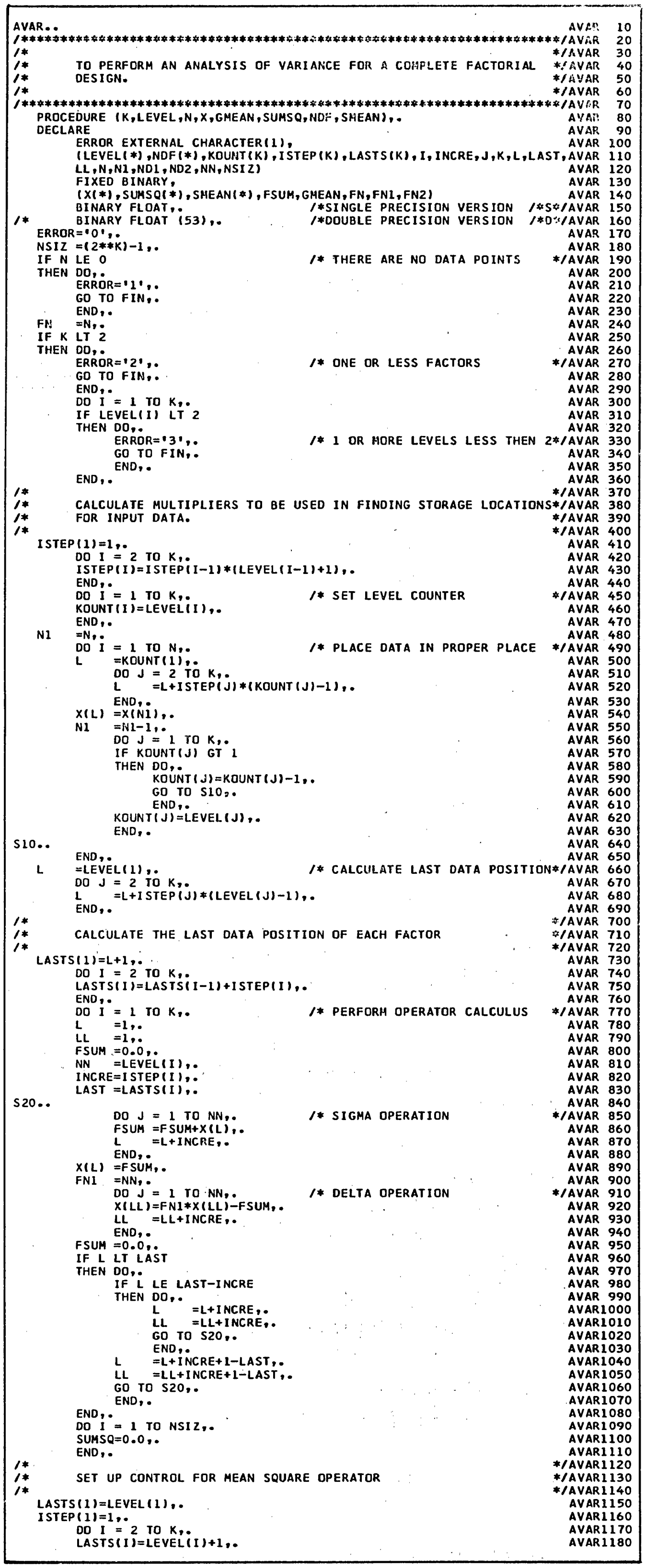
$$a_j = \frac{\sum_{k=1}^{q} t_{jk}\ b_k}{CANR} \tag{9}$$

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

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

Analysis of Variance

• Subroutine AVAR



```
AVAR..
/***********************************************************************/
/*                                                                     */
/*       TO PERFORM AN ANALYSIS OF VARIANCE FOR A COMPLETE FACTORIAL  */
/*       DESIGN.                                                       */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (K,LEVEL,N,X,GMEAN,SUMSQ,NDF,SMEAN),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),
        (LEVEL(*),NDF(*),KOUNT(K),ISTEP(K),LASTS(K),I,INCRE,J,K,L,LAST,
        LL,N,N1,ND1,ND2,NN,NSIZ)
        FIXED BINARY,
        (X(*),SUMSQ(*),SMEAN(*),FSUM,GMEAN,FN,FN1,FN2)
        BINARY FLOAT,.                    /*SINGLE PRECISION VERSION  /*S*/
/*      BINARY FLOAT (53),.               /*DOUBLE PRECISION VERSION  /*D*/
   ERROR='0',.
   NSIZ =(2**K)-1,.
   IF N LE 0                              /* THERE ARE NO DATA POINTS    */
   THEN DO,.
        ERROR='1',.
        GO TO FIN,.
        END,.
   FN   =N,.
   IF K LT 2
   THEN DO,.
        ERROR='2',.                       /* ONE OR LESS FACTORS         */
        GO TO FIN,.
        END,.
        DO I = 1 TO K,.
        IF LEVEL(I) LT 2
        THEN DO,.
             ERROR='3',.                  /* 1 OR MORE LEVELS LESS THEN 2*/
             GO TO FIN,.
             END,.
        END,.
/*                                                                     */
/*      CALCULATE MULTIPLIERS TO BE USED IN FINDING STORAGE LOCATIONS*/
/*      FOR INPUT DATA.                                                */
/*                                                                     */
   ISTEP(1)=1,.
        DO I = 2 TO K,.
        ISTEP(I)=ISTEP(I-1)*(LEVEL(I-1)+1),.
        END,.
        DO I = 1 TO K,.                   /* SET LEVEL COUNTER           */
        KOUNT(I)=LEVEL(I),.
        END,.
   N1   =N,.
        DO I = 1 TO N,.                   /* PLACE DATA IN PROPER PLACE  */
        L    =KOUNT(1),.
             DO J = 2 TO K,.
             L    =L+ISTEP(J)*(KOUNT(J)-1),.
             END,.
        X(L) =X(N1),.
        N1   =N1-1,.
             DO J = 1 TO K,.
             IF KOUNT(J) GT 1
             THEN DO,.
                  KOUNT(J)=KOUNT(J)-1,.
                  GO TO S10,.
                  END,.
             KOUNT(J)=LEVEL(J),.
             END,.
S10..
        END,.
   L    =LEVEL(1),.                       /* CALCULATE LAST DATA POSITION*/
        DO J = 2 TO K,.
        L    =L+ISTEP(J)*(LEVEL(J)-1),.
        END,.
/*                                                                     */
/*      CALCULATE THE LAST DATA POSITION OF EACH FACTOR               */
/*                                                                     */
   LASTS(1)=L+1,.
        DO I = 2 TO K,.
        LASTS(I)=LASTS(I-1)+ISTEP(I),.
        END,.
        DO I = 1 TO K,.                   /* PERFORM OPERATOR CALCULUS   */
        L    =1,.
        LL   =1,.
        FSUM =0.0,.
        NN   =LEVEL(I),.
        INCRE=ISTEP(I),.
        LAST =LASTS(I),.
S20..
             DO J = 1 TO NN,.             /* SIGMA OPERATION             */
             FSUM =FSUM+X(L),.
             L    =L+INCRE,.
             END,.
        X(L) =FSUM,.
        FN1  =NN,.
             DO J = 1 TO NN,.             /* DELTA OPERATION             */
             X(LL)=FN1*X(LL)-FSUM,.
             LL   =LL+INCRE,.
             END,.
        FSUM =0.0,.
        IF L LT LAST
        THEN DO,.
             IF L LE LAST-INCRE
             THEN DO,.
                  L    =L+INCRE,.
                  LL   =LL+INCRE,.
                  GO TO S20,.
                  END,.
             L    =L+INCRE+1-LAST,.
             LL   =LL+INCRE+1-LAST,.
             GO TO S20,.
             END,.
        END,.
        DO I = 1 TO NSIZ,.
        SUMSQ=0.0,.
        END,.
/*                                                                     */
/*      SET UP CONTROL FOR MEAN SQUARE OPERATOR                       */
/*                                                                     */
   LASTS(1)=LEVEL(1),.
   ISTEP(1)=1,.
        DO I = 2 TO K,.
        LASTS(I)=LEVEL(I)+1,.
```

```
       ISTEP(I)=ISTEP(I-1)*2,.                                          AVAR1190
       END,.                                                            AVAR1200
   NN  =1,.                                                             AVAR1210
       DO I = 1 TO K,.                                                  AVAR1220
       KOUNT(I)=0.0,.                                                   AVAR1230
       END,.                                                            AVAR1240
S30..                                                                   AVAR1250
   L   =0,.                                                             AVAR1260
       DO I = 1 TO K,.                                                  AVAR1270
       IF KOUNT(I) NE LASTS(I)                                          AVAR1280
       THEN DO,.                                                        AVAR1290
            IF L LE 0                                                   AVAR1300
            THEN DO,.                                                   AVAR1310
                 KOUNT(I)=KOUNT(I)+1,.                                  AVAR1320
                 IF KOUNT(I) LE LEVEL(I)                                AVAR1330
                 THEN GO TO S40,.                                       AVAR1340
                 GO TO S50,.                                            AVAR1350
                 END,.                                                  AVAR1360
            IF KOUNT(I)= LEVEL(I)                                       AVAR1370
            THEN GO TO S60,.                                            AVAR1380
S40..                                                                   AVAR1390
            L    =L+ISTEP(I),.                                          AVAR1400
            GO TO S60,.                                                 AVAR1410
            END,.                                                       AVAR1420
S50..                                                                   AVAR1430
       KOUNT(I)=0,.                                                     AVAR1440
S60..                                                                   AVAR1450
       END,.                                                            AVAR1460
   IF L GT 0                                                            AVAR1470
   THEN DO,.                                                            AVAR1480
       SUMSQ(L)=SUMSQ(L)+X(NN)*X(NN),.                                  AVAR1490
       NN   =NN+1,.                                                     AVAR1500
       GO TO S30,.                                                      AVAR1510
       END,.                                                            AVAR1520
   GMEAN=X(NN)/FN,.                          /* CALCULATE MEAN         */AVAR1530
/*                                                                     */AVAR1540
/*     CALCULATE FIRST DIVISOR REQUIRED TO FORM SUM OF SQUARES AND     */AVAR1550
/*     DIVISOR, WHICH IS EQUAL TO DEGREES OF FREEDOM, REQUIRED TO      */AVAR1560
/*     FORM MEAN SQUARES                                               */AVAR1570
/*                                                                     */AVAR1580
   ISTEP=0,.                                                            AVAR1590
   ISTEP(1)=1,.                                                         AVAR1600
   NN   =0,.                                                            AVAR1610
S70..                                                                   AVAR1620
   ND1  =1,.                                                            AVAR1630
   ND2  =1,.                                                            AVAR1640
       DO I = 1 TO K,.                                                  AVAR1650
       IF ISTEP(I) NE 0                                                 AVAR1660
       THEN DO,.                                                        AVAR1670
            ND1  =ND1*LEVEL(I),.                                        AVAR1680
            ND2  =ND2*(LEVEL(I)-1),.                                    AVAR1690
            END,.                                                       AVAR1700
       END,.                                                            AVAR1710
   FN1  =N*ND1,.                                                        AVAR1720
   FN2  =ND2,.                                                          AVAR1730
   NN   =NN+1,.                                                         AVAR1740
   SUMSQ(NN)=SUMSQ(NN)/FN1,.                                            AVAR1750
   SMEAN(NN)=SUMSQ(NN)/FN2,.                                            AVAR1760
   NDF(NN)=ND2,.                                                        AVAR1770
   IF NN LT LL                                                          AVAR1780
   THEN DO,.                                                            AVAR1790
            DO I = 1 TO K,.                                             AVAR1800
            IF ISTEP(I) NE 0                                            AVAR1810
            THEN ISTEP(I)=0,.                                           AVAR1820
            ELSE DO,.                                                   AVAR1830
                 ISTEP(I)=1,.                                           AVAR1840
                 GO TO S70,.                                            AVAR1850
                 END,.                                                  AVAR1860
            END,.                                                       AVAR1870
       END,.                                                            AVAR1880
FIN..                                                                   AVAR1890
   RETURN,.                                                             AVAR1900
   END,.                                     /*END OF PROCEDURE AVAR   */AVAR1910
```

Purpose:

AVAR performs an analysis of variance for a complete factorial design.

Usage:

CALL AVAR (K, LEVEL, N, X, GMEAN, SUMSQ, NDF, SMEAN);

Description of parameters:

K - BINARY FIXED
Given number of variables (factors).

LEVEL (K) - BINARY FIXED
Given vector, the i-th element being the number of levels for the i-th factor ($LEVEL_i$).

N - BINARY FIXED
Given total number of data points read in (N = [2 **K] -1).

X - BINARY FLOAT[(53)]
Given vector of length

$$\prod_{i=i}^{K} (LEVEL_i + 1)$$

with data positioned in locations one to N, where N is the total number of data points read in. The length of the vector must not exceed 32, 767.

GMEAN - BINARY FLOAT [(53)]
Resultant variable containing grand mean.

SUMSQ - BINARY FLOAT [(53)]
Resultant vector of length 2 to the K^{th} power minus one, ([2**K] - 1), containing the sums of squares.

NDF - BINARY FIXED
Resultant vector of length ([2**K] -1), containing degrees of freedom.

SMEAN - BINARY FLOAT [(53)]
Resultant vector of length ([2**K] - 1), containing mean squares.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - N, the number of data points, less than or equal to zero.
ERROR=2 - There is only one factor or less than one.
ERROR=3 - One or more factors have levels less than two.

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.

Mathematical Background:

This procedure calculates an analysis of variance in three steps:

1. The data is placed in properly distributed positions of storage.

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

$$MM = \prod_{i=1}^{K} (L_i + 1) \qquad (1)$$

where:

L_i = number of levels of i-th factor
K = number of factors

The data is redistributed according to equation (4) below. Prior to that, multipliers, S_j, to be used in finding proper positions of storage, are calculated as follows:

$$S_1 = 1 \qquad (2)$$

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

where j = 2, 3, ..., K.

Then the position to place each data point is calculated by the following equation:

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

where $KOUNT_j$ = value of the j-th subscript of the data to be stored. The procedure increments the value(s) of subscript(s) after each data point is stored.

2. The next step performs the calculus for the general K-factor experiment: operator Σ and operator Δ. An example is presented in terms of K = 3 to illustrate these operators.

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

With regard to the factor, A:

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

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

In mathematical notations, these operators are defined as follows:

$$\sum_a X_{abc} \equiv X_{.bc} \equiv \sum_{a=1}^{A} X_{abc} \qquad (5)$$

$$\Delta_a X_{abc} \equiv AX_{abc} - X_{.bc} \qquad (6)$$

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

3. In the next and final step the mean square operation for the general K-factor experiment is performed as follows:

a. Square each value of deviate for analysis of variance stored in array X, which is the result of the operators Σ and Δ applied in step 2.

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

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

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

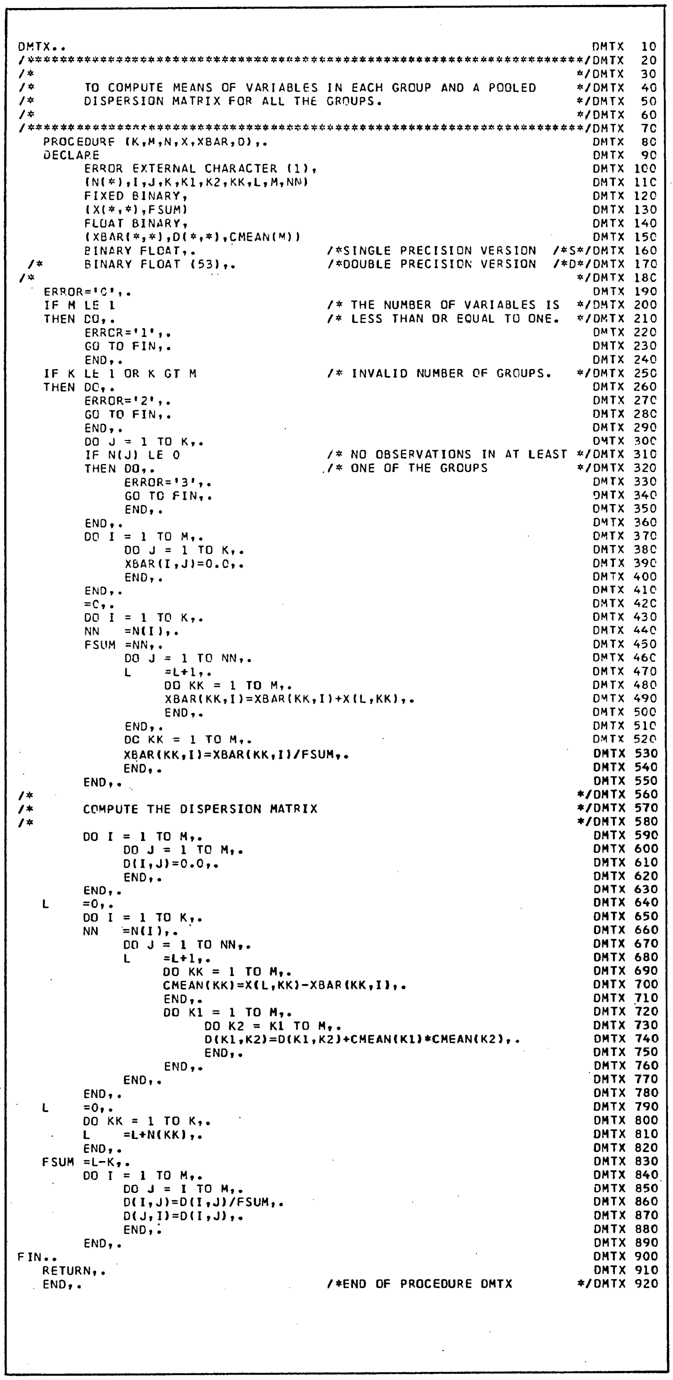
Designation of store and of quantity contained in it	Divisor required to form sum of squares of analysis of variance	Degrees of freedom required to form mean squares
$(A)^2$	ABC. A	(A-1)
$(B)^2$	ABC. B	(B-1)
$(AB)^2$	ABC. AB	(A-1) (B-1)
$(C)^2$	ABC. C	(C-1)
$(AC)^2$	ABC. AC	(A-1) (C-1)
$(BC)^2$	ABC. BC	(B-1) (C-1)
$(ABC)^2$	ABC. ABC	(A-1) (B-1) (C-1)

Programming Considerations:

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

Discriminant Analysis

● Subroutine DMTX



```
DMTX..
/**********************************************************************/
/*                                                                    */
/*      TO COMPUTE MEANS OF VARIABLES IN EACH GROUP AND A POOLED      */
/*      DISPERSION MATRIX FOR ALL THE GROUPS.                         */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (K,M,N,X,XBAR,D),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (N(*),I,J,K,K1,K2,KK,L,M,NN)
        FIXED BINARY,
        (X(*,*),FSUM)
        FLOAT BINARY,
        (XBAR(*,*),D(*,*),CMEAN(M))
        BINARY FLOAT,.                    /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.               /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
   ERROR='0',.
   IF M LE 1                              /* THE NUMBER OF VARIABLES IS  */
   THEN DO,.                              /* LESS THAN OR EQUAL TO ONE.  */
        ERROR='1',.
        GO TO FIN,.
        END,.
   IF K LE 1 OR K GT M                    /* INVALID NUMBER OF GROUPS.   */
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
        DO J = 1 TO K,.
        IF N(J) LE 0                      /* NO OBSERVATIONS IN AT LEAST */
        THEN DO,.                         /* ONE OF THE GROUPS           */
             ERROR='3',.
             GO TO FIN,.
             END,.
        END,.
        DO I = 1 TO M,.
             DO J = 1 TO K,.
             XBAR(I,J)=0.0,.
             END,.
        END,.
        =0,.
        DO I = 1 TO K,.
        NN   =N(I),.
        FSUM =NN,.
             DO J = 1 TO NN,.
             L    =L+1,.
                  DO KK = 1 TO M,.
                  XBAR(KK,I)=XBAR(KK,I)+X(L,KK),.
                  END,.
             END,.
             DO KK = 1 TO M,.
             XBAR(KK,I)=XBAR(KK,I)/FSUM,.
             END,.
        END,.
/*                                                                    */
/*      COMPUTE THE DISPERSION MATRIX                                 */
/*                                                                    */
        DO I = 1 TO M,.
             DO J = 1 TO M,.
             D(I,J)=0.0,.
             END,.
        END,.
   L    =0,.
        DO I = 1 TO K,.
        NN   =N(I),.
             DO J = 1 TO NN,.
             L    =L+1,.
                  DO KK = 1 TO M,.
                  CMEAN(KK)=X(L,KK)-XBAR(KK,I),.
                  END,.
                  DO K1 = 1 TO M,.
                       DO K2 = K1 TO M,.
                       D(K1,K2)=D(K1,K2)+CMEAN(K1)*CMEAN(K2),.
                       END,.
                  END,.
             END,.
        END,.
   L    =0,.
        DO KK = 1 TO K,.
        L    =L+N(KK),.
        END,.
   FSUM =L-K,.
        DO I = 1 TO M,.
             DO J = I TO M,.
             D(I,J)=D(I,J)/FSUM,.
             D(J,I)=D(I,J),.
             END,.
        END,.
FIN..
   RETURN,.
   END,.                                  /*END OF PROCEDURE DMTX        */
```

Purpose:

DMTX computes means of variables in each group and a pooled dispersion matrix for all the groups. This subroutine is used in the performance of discriminant analysis.

Usage:

CALL DMTX (K, M, N, X, XBAR, D);

K - BINARY FIXED
Given number of groups. K must be greater than 1.

M - BINARY FIXED
Given number of variables (must be the same for all groups).

N(K) - BINARY FIXED
Given vector containing sample sizes of groups. $N=(n_1, n_2, \ldots, n_k)$

X(NN, M) BINARY FLOAT
Given matrix containing data in a manner equivalent to a three-dimensional array (X_{ijk}). The first subscript is case number; the second, variable number; the third, group number. $NN=n_1 + n_2 + \ldots + n_k$.

XBAR(M, K) - BINARY FLOAT [(53)]
Resultant matrix containing means of variables in K groups.

D(M, M) - BINARY FLOAT [(53)]
Resultant matrix containing pooled dispersion.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of variables less than or equal to one.
ERROR=2 - invalid number of groups ($K \le 1$ or $K > M$).
ERROR=3 - no observations in one or more groups.

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

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, Sections 6.6-6.8.

Mathematical Background:

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, ..., K, the subroutine calculates means and sums of cross-products of deviations from means as shown below.

Means:

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

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

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

Sum of cross-products of deviations from means:

$$S_k = \left\{ s_{j\,l}^{\,k} \right\} = \sum (x_{ijk} - \bar{x}_{jk})\ (x_{ilk} - \bar{x}_{lk}) \tag{2}$$

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

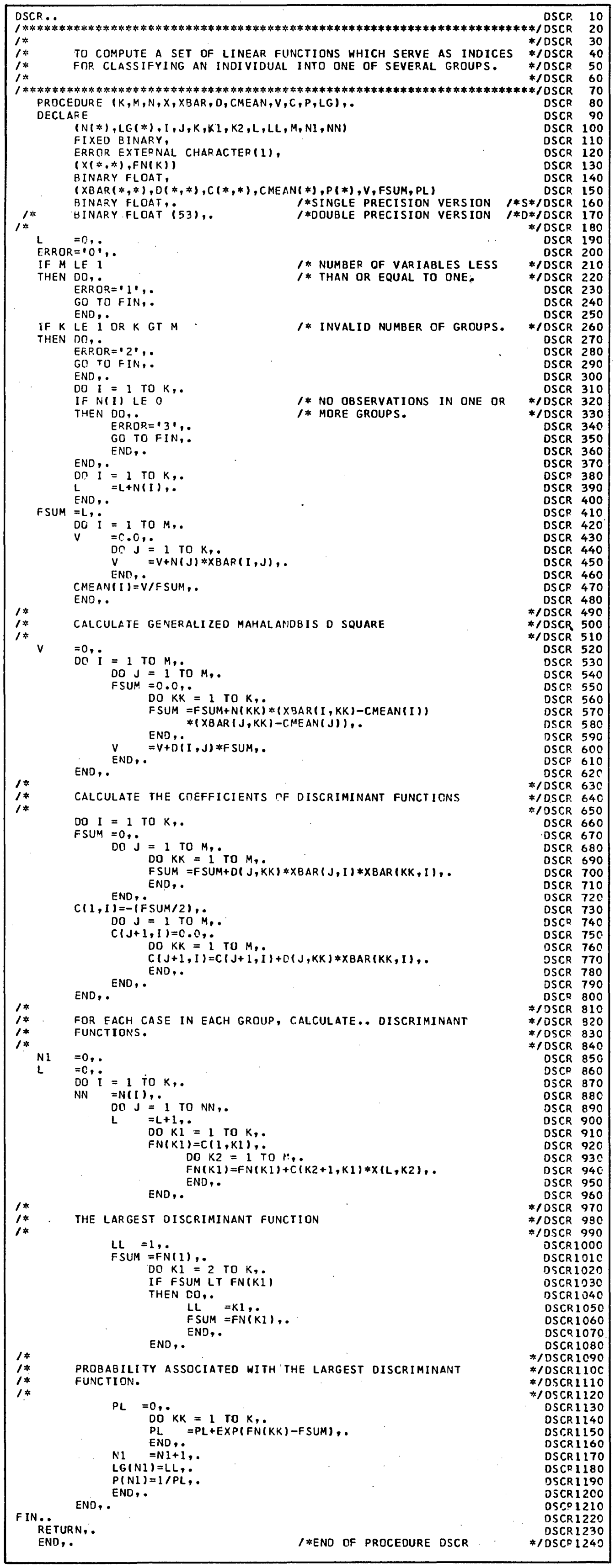
l = 1, 2, ..., m

The pooled dispersion matrix is calculated as follows:

$$D = \frac{\sum_{k=1}^{K} S_k}{\sum_{k=1}^{K} n_k - K} \tag{3}$$

where K = number of groups

- Subroutine DSCR



```
DSCR..
/***********************************************************************/
/*                                                                     */
/*      TO COMPUTE A SET OF LINEAR FUNCTIONS WHICH SERVE AS INDICES   */
/*      FOR CLASSIFYING AN INDIVIDUAL INTO ONE OF SEVERAL GROUPS.     */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (K,M,N,X,XBAR,D,CMEAN,V,C,P,LG),.
   DECLARE
        (N(*),LG(*),I,J,K,K1,K2,L,LL,M,N1,NN)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (X(*,*),FN(K))
        BINARY FLOAT,
        (XBAR(*,*),D(*,*),C(*,*),CMEAN(*),P(*),V,FSUM,PL)
        BINARY FLOAT,.                /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.           /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                     */
   L    =0,.
   ERROR='0',.
   IF M LE 1                          /* NUMBER OF VARIABLES LESS      */
   THEN DO,.                          /* THAN OR EQUAL TO ONE.         */
        ERROR='1',.
        GO TO FIN,.
        END,.
   IF K LE 1 OR K GT M                /* INVALID NUMBER OF GROUPS.     */
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
        DO I = 1 TO K,.
        IF N(I) LE 0                  /* NO OBSERVATIONS IN ONE OR     */
        THEN DO,.                     /* MORE GROUPS.                  */
             ERROR='3',.
             GO TO FIN,.
             END,.
        END,.
        DO I = 1 TO K,.
        L    =L+N(I),.
        END,.
   FSUM =L,.
        DO I = 1 TO M,.
        V    =0.0,.
             DO J = 1 TO K,.
             V    =V+N(J)*XBAR(I,J),.
             END,.
        CMEAN(I)=V/FSUM,.
        END,.
/*                                                                     */
/*      CALCULATE GENERALIZED MAHALANDBIS D SQUARE                    */
/*                                                                     */
   V    =0,.
        DO I = 1 TO M,.
             DO J = 1 TO M,.
             FSUM =0.0,.
                  DO KK = 1 TO K,.
                  FSUM =FSUM+N(KK)*(XBAR(I,KK)-CMEAN(I))
                       *(XBAR(J,KK)-CMEAN(J)),.
                  END,.
             V    =V+D(I,J)*FSUM,.
             END,.
        END,.
/*                                                                     */
/*      CALCULATE THE COEFFICIENTS OF DISCRIMINANT FUNCTIONS          */
/*                                                                     */
        DO I = 1 TO K,.
        FSUM =0,.
             DO J = 1 TO M,.
                  DO KK = 1 TO M,.
                  FSUM =FSUM+D(J,KK)*XBAR(J,I)*XBAR(KK,I),.
                  END,.
             END,.
        C(1,I)=-(FSUM/2),.
             DO J = 1 TO M,.
             C(J+1,I)=0.0,.
                  DO KK = 1 TO M,.
                  C(J+1,I)=C(J+1,I)+D(J,KK)*XBAR(KK,I),.
                  END,.
             END,.
        END,.
/*                                                                     */
/*      FOR EACH CASE IN EACH GROUP, CALCULATE.. DISCRIMINANT         */
/*      FUNCTIONS.                                                     */
/*                                                                     */
   N1   =0,.
   L    =0,.
        DO I = 1 TO K,.
        NN   =N(I),.
             DO J = 1 TO NN,.
             L    =L+1,.
                  DO K1 = 1 TO K,.
                  FN(K1)=C(1,K1),.
                       DO K2 = 1 TO M,.
                       FN(K1)=FN(K1)+C(K2+1,K1)*X(L,K2),.
                       END,.
                  END,.
/*                                                                     */
/*      THE LARGEST DISCRIMINANT FUNCTION                             */
/*                                                                     */
             LL   =1,.
             FSUM =FN(1),.
                  DO K1 = 2 TO K,.
                  IF FSUM LT FN(K1)
                  THEN DO,.
                       LL   =K1,.
                       FSUM =FN(K1),.
                       END,.
                  END,.
/*                                                                     */
/*      PROBABILITY ASSOCIATED WITH THE LARGEST DISCRIMINANT          */
/*      FUNCTION.                                                      */
/*                                                                     */
             PL   =0,.
                  DO KK = 1 TO K,.
                  PL   =PL+EXP(FN(KK)-FSUM),.
                  END,.
             N1   =N1+1,.
             LG(N1)=LL,.
             P(N1)=1/PL,.
             END,.
        END,.
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE DSCR          */
```

Purpose:

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

Usage:

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

K - BINARY FIXED
Given number of groups. K must be greater than 1.

M - BINARY FIXED
Given number of variables.

N(K) - BINARY FIXED
Given vector containing sample sizes of groups.
$N = (n_1, n_2, \ldots, n_K)$

X(NN, M) - BINARY FLOAT
Given matrix containing data in the manner equivalent to a three-dimensional array $\{X_{ijk}\}$. The first subscript is case number; the second, variable number; the third, group number. $NN = n_1 + n_2 + \ldots + n_k$.

XBAR(M,K) - BINARY FLOAT [(53)]
Given matrix containing means of M variables in K groups.

D(M, M) - BINARY FLOAT [(53)]
Given matrix containing the inverse of pooled dispersion matrix.

CMEAN(M) - BINARY FLOAT [(53)]
Resultant vector containing common means.

V - BINARY FLOAT [(53)]
Resultant variable containing generalized Mahalanobis D-square.

C(M+1,K) - BINARY FLOAT [(53)]
Resultant matrix containing the coefficients of discriminant functions. The first position of each column (function) contains the value of the constant for that function.

P(NN) - BINARY FLOAT [(53)]
Resultant 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 two-dimensional array (the first subscript is case number, and the second subscript is group number).
$NN=n_1+n_2+\ldots+n_K$

LG(NN) - BINARY FIXED
Resultant vector containing the subscripts of the largest discriminant functions stored in vector P.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of variables less than or equal to one.
ERROR=2 - invalid number of groups ($K \leq 1$ or $K > M$).
ERROR=3 - no observations in one or more groups.

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

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.

Mathematical Background:

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:

$$\bar{x}_j = \frac{\sum\limits_{k=1}^{K} n_k \bar{x}_{jk}}{\sum\limits_{k=1}^{K} n_k} \tag{1}$$

where:

K = number of groups

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

n_k = sample size in the k-th group

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

Generalized Mahalanobis D^2 statistics, V:

$$V = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \sum_{k=1}^{K} a_{ijk}$$

$$a_{ijk} = n_k (\bar{x}_{ik} - \bar{X}_i)(\bar{x}_{jk} - \bar{X}_j) \qquad (2)$$

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 (K-1) degrees of freedom to test the hypothesis that the mean values are the same in all the K groups for these m variables. For each discriminant function $k_* = 1, 2, \ldots, K$, the following statistics are calculated.

Coefficients:

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

where:

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

$k = k_*$

Constant:

$$C_{0k*} = -1/2 \sum_{j=1}^{m} \sum_{l=1}^{m} d_{jl} \bar{x}_{jk} \bar{x}_{lk} \qquad (4)$$

For each i-th case in each k-th group, the following calculations are performed.

Discriminant functions:

$$f_{k*} = \sum_{j=1}^{m} C_{jk} x_{ijk} + C_{0k*} \qquad (5)$$

where:

$k_* = 1, 2, \ldots, K$

Probability associated with largest discriminant function:

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

where:

f_L = the value of the largest discriminant function

L = the subscript of the largest discriminant function

Principal Components Analysis

- Subroutine TRAC

```
TRAC..
/************************************************************************/
/*                                                                      */
/*      TO COMPUTE CUMULATIVE PERCENTAGE OF EIGENVALUES GREATER         */
/*      THAN OR EQUAL TO A CONSTANT SPECIFIED BY THE USER.              */
/*                                                                      */
/************************************************************************/
   PROCEDURE (M,R,CON,K,D),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (I,J,K,M)
        FIXED BINARY,
        (R(*,*),D(*),CON)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLCAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                      */
   ERROR='0',.
   IF M LE 0                           /* ORDER OF MATRIX IS ZERO.      */
   THEN DO,.
        ERROR='1',.
        GO TO S20,.
        END,.
        DO I = 1 TO M,.
        END,.
   K    =C,.
/*                                                                      */
/*      TEST WHETHER I-TH EIGENVALUE IS GREATER THAN OR EQUAL TO        */
/*      THE CONSTANT.                                                   */
/*                                                                      */
        DO I = 1 TO M,.
        IF D(I) LT CON
        THEN GO TO S1C,.
        K    =K+1,.
        D(I) =D(I)/M,.
        END,.
S10..
   IF K LE 1
   THEN DO,.
        ERROR='2',.                    /* NOT ENOUGH EIGENVALUES        */
        GO TO S2C,.                    /* ARE RETAINED                  */
        END,.
        DO I = 2 TO K,.
        D(I) =D(I)+D(I-1),.
        END,.
S2C..
   RETURN,.
   END,.                               /*END OF PROCEDURE TRAC          */
```

Purpose:

TRAC computes cumulative percentage of eigenvalues greater than or equal to a constant specified by the user.

Usage:

CALL TRAC (M, R, CON, K, D);

Description of parameters:

M - BINARY FIXED
Given number of variables.

R(M, M) - BINARY FLOAT [(53)]
Given matrix containing eigenvalues in diagonal. Eigenvalues are assumed to be arranged in descending order.

CON - BINARY FLOAT [(53)]
Given constant used to decide how many eigenvalues to retain. Cumulative percentage of eigenvalues greater than or equal to this value is calculated.

K - BINARY FIXED
Resultant variable containing the number of eigenvalues greater than or equal to CON. (K is the number of factors.)

D(M) - BINARY FLOAT [(53)]
Resultant vector containing cumulative percentage of eigenvalues greater than or equal to CON.

Remark:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - order of matrix equal to zero.
ERROR=2 - number of eigenvalues retained less than or equal to one.

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.

Mathematical Background:

This procedure finds K, the number of eigenvalues greater than or equal to the value of a special constant. The given eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ must be arranged in descending order.

Cumulative percentages for those K eigenvalues are:

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

where:

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

M = number of eigenvalues (or variables)

$K \leq M$

- Subroutine LOAD

```
LOAD..
/*********************************************************************/
/*                                                                   */
/*      TO COMPUTE A FACTOR MATRIX (LOADING) FROM EIGENVALUES AND    */
/*      ASSOCIATED EIGENVECTORS.                                     */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (M,K,R,V),.
   DECLARE
        (I,J,K)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (R(*,*),V(*,*),SQ)
        BINARY FLOAT,.                  /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.             /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                   */
   ERROR='0',.
   IF K LE 1 OR K GT M                  /* INVALID VALUE OF K        */
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
   IF M LE 0                            /* ORDER OF MATRIX IS ZERO   */
   THEN ERROR='1',.
   ELSE DO,.
            DO J = 1 TO K,.
            SQ   =SQRT(R(J,J)),.
                 DO I = 1 TO M,.
                 V(I,J)=SQ*V(I,J),.
                 END,.
            END,.
        END,.
FIN..
   RETURN,.
   END,.                                /*END OF PROCEDURE LOAD      */
```

Purpose:

LOAD computes a factor matrix (loading) from eigenvalues and associated eigenvectors.

Usage:

CALL LOAD (M, K, R, V);

Description of parameters:

M -	BINARY FIXED Given number of variables.
K -	BINARY FIXED Given number of factors.
R(M, M) -	BINARY FLOAT [(53)] Given matrix containing eigenvalues in the diagonal. Eigenvalues are assumed to be arranged in descending order. The first K eigenvalues are used by this procedure.
V(M, M) -	BINARY FLOAT [(53)] Given matrix V contains eigenvectors columnwise. Resultant matrix V contains a factor matrix (M by K).

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - the order of the matrix is zero.
ERROR=2 - invalid number of factors ($K \leq 1$ or $K > M$).

Method:

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

Mathematical Background:

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

$$a_{ij} = V_{ij} \cdot \sqrt{\lambda_j}$$

where:

$i = 1, 2, \ldots, M$ are indices of variables

$j = 1, 2, \ldots, K$ are indices of eigenvalues retained (see the subroutine TRAC)

$K \leq M$

• Subroutine VRMX

```
VRMX..
/**********************************************************************/
/*                                                                    */
/*      TO PERFORM ORTHOGONAL ROTATION OF A FACTOR MATRIX.            */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,K,A,NC,TV,H,F,D),.
   DECLARE
        (I,II,J,K,K1,LL,M,NC,NV)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        (A(*,*),TV(*),H(*),F(*),D(*),EPS,TVLT,FN,AA,BB,CC,DD,G,B,U,T,
        COS4T,SIN4T,TAN4T,SINP,COSP,CTN4T,COS2T,SIN2T,COST,SINT,CONS)
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/
 /*     BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/
/*                                                                    */
   EPS  =.00116,.                      /* INITIALIZATION              */
   TVLT =0,.
   LL   =K-1,.
   NV   =1,.
   NC   =0,.
   FN   =M*M,.
   CONS =.7071066,.
   ERROR='0',.
   IF M LE 1                           /* NUMBER OF VARIABLES LESS    */
   THEN DO,.                           /* THAN OR EQUAL TO ONE        */
        ERROR='1',.
        GO TO FIN,.
        END,.
   IF K LE 1 OR K GT M                 /* INVALID VALUE OF K          */
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
/*                                                                    */
/*      CALCULATE ORIGINAL COMMUNALITIES                              */
/*                                                                    */
        DO I = 1 TO M,.
        H(I) =0,.
             DO J = 1 TO K,.
             H(I) =H(I)+A(I,J)*A(I,J),.
             END,.
        END,.
/*                                                                    */
/*      CALCULATE NORMALIZED FACTOR MATRIX                            */
/*                                                                    */
        DO I = 1 TO M,.
        H(I) =SQRT(H(I)),.
             DO J = 1 TO K,.
             A(I,J)=A(I,J)/H(I),.
             END,.
        END,.
   GO TO S20,.
/*                                                                    */
/*      CALCULATE VARIANCE FOR FACTOR MATRIX                          */
/*                                                                    */
S10..
   NV   =NV+1,.
   TVLT =TV(NV-1),.
S20..
   TV(NV)=0,.
        DO J = 1 TO K,.
        AA   =0,.
        BB   =0,.
             DO I = 1 TO M,.
             CC   =A(I,J)*A(I,J),.
             AA   =AA+CC,.
             BB   =BB+CC*CC,.
             END,.
        TV(NV)=TV(NV)+(M*BB-AA*AA)/FN,.
        END,.
   IF NV GE 51
   THEN DO,.                           /* NUMBER OF ITERATIONS = 50   */
        ERROR='3',.
        GO TO S80,.
        END,.
   IF TV(NV)-TVLT LE 1.0E-7            /* PERFORM CONVERGENCE TEST    */
   THEN DO,.
        NC   =NC+1,.
        IF NC GT 3
        THEN GO TO S80,.
        END,.
/*                                                                    */
/*      ROTATION OF TWO FACTORS BEGINS                                */
/*                                                                    */
        DO J = 1 TO LL,.
        II   =J+1,.
             DO K1 = II TO K,.
             AA   =0,.                 /* CALCULATE NUM AND DEN       */
             BB   =0,.
             CC   =0,.
             DD   =0,.
                  DO I = 1 TO M,.
                  U    =(A(I,J)+A(I,K1))*(A(I,J)-A(I,K1)),.
                  T    =A(I,J)*A(I,K1)*2,.
                  CC   =CC+(U+T)*(U-T),.
                  DD   =DD+2*U*T,.
                  AA   =AA+U,.
                  BB   =BB+T,.
                  END,.
             T    =DD-2*AA*BB/M,.
             B    =CC-(AA*AA-BB*BB)/M,.
             IF T = B
             THEN DO,.
                  IF T+B LT EPS
                  THEN GO TO S70,.
/*                                                                    */
/*      NUM + DEN IS GREATER THAN OR EQUAL TO THE TOLERANCE FACTOR    */
/*                                                                    */
             COS4T=CONS,.
                  SIN4T=CONS,.
                  GO TO S40,.
                  END,.
             IF T GT B
             THEN GO TO S30,.
             TAN4T=ABS(T)/ABS(B),.     /* NUM IS LESS THAN DEN        */
             IF TAN4T GE EPS
             THEN DO,.
                  COS4T=1/SQRT(1+TAN4T*TAN4T),.
                  SIN4T=TAN4T*COS4T,.
                  GO TO S40,.
                  END,.
             IF B GE 0
             THEN GO TO S70,.
             SINP =CONS,.
             COSP =CONS,.
             GO TO S60,.
S30..
             CTN4T=ABS(T/B),.          /* NUM IS GREATER THAN DEN     */
             IF CTN4T GE EPS
             THEN DO,.
                  SIN4T=1/SQRT(1+CTN4T*CTN4T),.
                  COS4T=CTN4T*SIN4T,.
                  GO TO S40,.
                  END,.
             COS4T=0,.
             SIN4T=1,.
/*                                                                    */
/*      DETERMINE COS THEAT AND SIN THETA                             */
/*                                                                    */
S40..
             COS2T=SQRT((1+COS4T)/2),.
             SIN2T=SIN4T/(2*COS2T),.
             COST =SQRT((1+COS2T)/2),.
             SINT =SIN2T/(2*COST),.
/*                                                                    */
/*      DETERMINE COS PHI AND SIN PHI                                 */
/*                                                                    */
             IF B GT 0
             THEN DO,.
                  COSP =COST,.
                  SINP =SINT,.
                  GO TO S50,.
                  END,.
             COSP =CONS*(COST+SINT),.
             SINP =ABS(CONS*(COST-SINT)),.
S50..
             IF T LE 0
             THEN SINP =-SINP,.
S60..
                  DO I = 1 TO M,.      /* PERFORM ROTATION            */
                  AA   =A(I,J)*COSP+A(I,K1)*SINP,.
                  A(I,K1)=-A(I,J)*SINP+A(I,K1)*COSP,.
                  A(I,J)=AA,.
                  END,.
S70..
             END,.
        END,.
   GO TO S10,.
/*                                                                    */
/*      DENORMALIZE VARIMAX LOADINGS                                  */
/*                                                                    */
S80..
        DO I = 1 TO M,.
             DO J = 1 TO K,.
             A(I,J)=A(I,J)*H(I),.
             END,.
        END,.
   NC   =NV-1,.                        /* CHECK ON COMMUNALITIES      */
   H    =H*H,.
        DO I = 1 TO M,.
        F(I) =0,.
             DO J = 1 TO K,.
             F(I) =F(I)+A(I,J)*A(I,J),.
             END,.
        D(I) =H(I)-F(I),.
        END,.
FIN..
   RETURN,.
   END,.                               /*END OF PROCEDURE VRMX        */
```

Purpose:

VRMX performs an orthogonal rotation of a factor matrix.

Usage:

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

M - BINARY FIXED
Given number of variables.

K - BINARY FIXED
Given number of factors.

A(M,K) - BINARY FLOAT [(53)]
Given factor matrix.
Resultant rotated M x K factor matrix.

NC - BINARY FIXED
Resultant variable containing the number of iteration cycles performed.

TV(51) - BINARY FLOAT [(53)]
Resultant 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 procedure is 50.

H(M) - BINARY FLOAT [(53)]
Resultant vector containing the original communalities.

F(M) - BINARY FLOAT [(53)]
Resultant vector containing the final communalities.

D(M) - BINARY FLOAT [(53)]
Resultant vector containing the difference between the original and final communalities.

Remarks:

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

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of variables less than or equal to one.
ERROR=2 - invalid number of factors (K≤1 or K >M).
ERROR=3 - 50 iterations executed without convergence.

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.

Mathematical Background:

This subroutine performs orthogonal rotations on an 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, . . . , m are variables, j = 1, 2, . . . , k are factors, a_{ij} is the loading for the i-th variable on the j-th factor, and h_i^2 is the communality of the i-th variable defined below.

Communalities:

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

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

Normalized factor matrix:

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

where:

i = 1,2, . . . , m

j = 1,2, . . . , k

Variance for factor matrix:

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

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

Convergence test:

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

four successive times, the program stops rotation and performs 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.

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

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

where x_i and y_i are presently available normalized loadings, and X_i and Y_i, the desired normalized loadings, are functions of ϕ, the angle of rotation. The computational steps are 1 through 5 below:

1. Calculation of NUM and DEN:

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

$$B = 2 \sum_i x_i y_i$$

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

$$D = 4 \sum_i (x_i + y_i)(x_i - y_i) x_i y_i \tag{7}$$

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

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

2. Comparison of NUM and DEN:

The following four cases may arise.

NUM < DEN, go to (2a) below

NUM > DEN, go to (2b) below

(NUM + DEN) $\geq \epsilon$*, go to (2c) below

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

* ϵ is a small tolerance factor.

a. $\tan 4\theta = |\text{NUM}|/|\text{DEN}|$ (8)

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

DEN is positive, skip to the next rotation.

DEN is negative, set $\cos\phi = \sin\phi = (\sqrt{2})/2$ and go to step 5.

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

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

$$\sin 4\theta = \tan 4\theta \circ \cos 4\theta \tag{10}$$

and go to step 3.

b. $\text{ctn}\, 4\theta = |\text{NUM}|/|\text{DEN}|$ (11)

If $\text{ctn}\, 4\theta < \epsilon$, set $\cos 4\theta = 0$ and $\sin 4\theta = 1$. Go to step 3.

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

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

$$\cos 4\theta = \text{ctn}\, 4\theta \circ \sin 4\theta \tag{13}$$

and go to step 3.

c. Set $\cos 4\theta = \sin 4\theta = (\sqrt{2})/2$ and go to step 3.

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

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

$$\sin 2\theta = \sin 4\theta/2 \cos 2\theta \tag{15}$$

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

$$\sin\theta = \sin 2\theta/ 2 \cos\theta \tag{17}$$

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

a. If DEN is positive, set

$$\cos\phi = \cos\theta \tag{18}$$

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

and go to (4b).

If DEN is negative, calculate

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

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

and go to (4b).

b. If NUM is positive, set

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

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

and go to step 5.

If NUM is negative, set

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

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

5. Rotation:

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

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

where

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

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

Denormalization:

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

where:

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

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

Check on communalities:

Final communalities

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

Difference

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

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

Nonparametric Statistics

• Subroutine KLMO

```
KLMO..
/**********************************************************************/
/*                                                                    */
/*      TESTS THE DIFFERENCE BETWEEN EMPIRICAL AND THEORETICAL        */
/*      DISTRIBUTIONS USING THE KOLMOGOROV-SMIRNOV TEST.              */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,N,Z,PROB,IFCOD,U,S),.
   DECLARE
        (X(*),Y,TEMP,PROB,S,U,Z,D,DN,EI,ES,FI,FS) FLOAT BINARY,
        (I,J,IL,N,IFCOD) FIXED BINARY,
        ERROR EXTERNAL CHARACTER (1),.
   ERROR='0',.
   IF N LT 100
   THEN IF N=0                        /* N < 100--SET ERROR IND.      */
        THEN DO,.
             ERROR='4',.
             GO TO S80,.
             END,.
        ELSE ERROR='3',.
        DO I=1 TO N-1,.               /* SORT X INTO                  */
             DO J=I+1 TO N,.          /* ASCENDING SEQUENCE           */
             IF X(I) GT X(J)
             THEN DO,.
                  TEMP =X(I),.
                  X(I) =X(J),.
                  X(J) =TEMP,.
                  END,.
             END,.
        END,.
                                      /* COMPUTES MAX. DEV. DN IN     */
                                      /* ABS. VAL. BETWEEN EMP. AND   */
                                      /* THEO. FUNCTIONS OVER ALL X   */
   DN,FS=0.0,.
   IL   =1,.
S10..
        DO I=IL TO N-1,.
        J    =I,.
        IF X(J)=X(J+1)
        THEN GO TO S20,.
        ELSE GO TO S40,.
S20..
   END,.
S30..
   J    =N,.
S40..
   IL   =J+1,.
   FI   =FS,.
   FS   =FLOAT(J)/N,.                 /* EMP. DIST. FUNCT. CALCULATED*/
   IF IFCOD=2
   THEN DO,.
        IF S LE 0
        THEN
S50..
             DO,.                     /* INVALID VALUE OF S           */
             ERROR='1',.
             GO TO S80,.
             END,.
        ELSE DO,.                     /* EXPONENTIAL PDF              */
             Z    =(X(J)-U)/S+1.0,.
             IF Z LE 0
             THEN                     /* Z < OR = 0                   */
S60..
                  DO,.
                  Y    =0.0,.
S70..
                  EI   =ABS(Y-FI),.
                  ES   =ABS(Y-FS),.
                                      /* COMPUTE MAX. DEV. DN BETWEEN*/
                                      /* EMP. AND THEO. FUNCTIONS     */
                  DN   =MAX(DN,EI,ES),.
                  IF IL=N
                  THEN GO TO S30,.
                  ELSE IF IL LT N
                       THEN GO TO S10,.
                       ELSE DO,.
                                      /* CALC. ASYMPTOTIC VALUES      */
                                      /* USING SMIR                   */
                            Z    =DN*SQRT(N),.
                            CALL SMIR (Z,PROB),.
                            PROB=1.0E0-PROB,.
                            GO TO S80,.
                            END,.
                  END,.
             ELSE DO,.                /* EXPONENTIAL PDF              */
                  Y=1.-EXP(-Z),.
                  GO TO S70,.
                  END,.
             END,.
        END,.
   ELSE IF IFCOD LT 2
        THEN IF S LE 0
             THEN GO TO S50,.         /* INVALID VALUE OF S           */
             ELSE DO,.                /* NORMAL PDF                   */
                  Z    =(X(J)-U)/S,.
                  CALL NDTR(Z,Y,D),.
                  GO TO S70,.
                  END,.
        ELSE IF IFCOD=4
             THEN IF S LE U
                  THEN GO TO S50,.    /* INVALID VAL. OF S OR U       */
                  ELSE IF X(J) LE U   /* UNIFORM PDF                  */
                       THEN GO TO S6C,.
                       ELSE IF X(J) LE S
                            THEN DO,.
                                 Y    =(X(J)-U)/(S-U),.
                                 GO TO S70,.
                                 END,.
                            ELSE DO,.
                                 Y    =1.0,.
                                 GO TO S70,.
                                 END,.
             ELSE IF IFCOD LT 4
                  THEN IF S=0         /* INVALID VALUE OF S           */
                       THEN GO TO S50,.
                       ELSE DO,.      /* CAUCHY PDF                   */
                            Y    =ATAN((X(J)-U)/S)*0.3183099+0.5,.
                            GO TO S70,.
                            END,.
```

```
                ELSE ERROR='2',.    /* USER'S PDF                   */KLMO1200
S80..                                                                 KLMO1210
   RETURN,.                                                           KLMO1220
   END,.                            /*END OF PROCEDURE KLMO         */KLMO1230
```

Purpose:

KLMO tests the difference between empirical and theoretical distributions using the Kolmogorov-Smirnov test.

Usage:

CALL KLMO (X, N, Z, PROB, IFCOD, U, S);

X(N) - BINARY FLOAT
Given vector of independent observations.

N - BINARY FIXED
Given number of observations in X.

Z - BINARY FLOAT
Resultant variable containing the greatest value with respect to X of $\sqrt{N}$ ($|F_N(x) - F(x)|$), where F(x) is a theoretical distribution function and $F_N(x)$ is an empirical distribution function.

PROB - BINARY FLOAT
Resultant variable containing the probability of the statistic being greater than or equal to Z if the hypothesis that X is from the density under consideration is true. For example, PROB=0.05 implies that X can be considered to be from the density under consideration with 5% probability of being incorrect.
PROB=1. - SMIR (Z).

IFCOD - BINARY FIXED
Given code denoting the particular theoretical probability distribution function being considered. When IFCOD
=1, F(x) is the normal PDF
=2, F(x) is the exponential PDF
=3, F(x) is the Cauchy PDF
=4, F(x) is the uniform PDF
=5, F(x) is user-supplied.

U - BINARY FLOAT
When IFCOD is 1 or 2, U is the given mean of the density given above.
When IFCOD is 3, U is the given median of the Cauchy density.
When IFCOD is 4, U is the given left endpoint of the uniform density.
When IFCOD is 5, U is user-specified.

S - BINARY FLOAT
When IFCOD is 1 or 2, S is the given standard deviation of density given above, and should be positive.
When IFCOD is 3, (U-S) specifies the first quartile of the Cauchy density. S given should be nonzero.
If IFCOD is 4, S is the given right endpoint of the uniform density. S should be greater than U.
If IFCOD is 5, S is user-specified.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - invalid value of S (if IFCOD = 4, S or U is invalid).
ERROR=2 - requested user's PDF has not been supplied.
ERROR=3 - number of observations less than 100.
ERROR=4 - number of observations equal to zero.

N should be greater than or equal to 100 (see the mathematical background for subroutine SMIR, for the asymptotic formulae). Also, probability levels determined by this program will not be correct if the same samples used in this test are used to estimate parameters for the continuous distribution.

Any user-supplied cumulative probability distribution function should be coded beginning with program comments "USER'S PDF" and should return to S70.

Subroutines and function subroutines required:

SMIR
NDTR

Method:

For references see:

W. Feller, "On the Kolmogorov-Smirnov limit theorems for empirical distributions", Annals of Math. Stat., 19, pp. 177-189.

N. Smirnov, "Table for estimating the goodness of fit of empirical distributions", Annals of Math. Stat., 19, pp. 279-281.

R. Von Mises, Mathematical Theory of Probability and Statistics. Academic Press, New York, 1964, pp. 490-493.

B. V. Gnedenko, The Theory of Probability. Chelsea Publishing Co., New York, 1962, pp. 384-401.

H.W. Lilliefors, "On the Kolmogorov-Smirnov test for normality with mean and variance unknown", J.A.S.A., 62 (1967), pp. 399-402.

Mathematical Background:

Given a sample of n independent and identically distributed random variables $X_1, X_2, \ldots, X_n$ with continuous cumulative distribution function F(x), this subroutine tests the difference in absolute value between the empirical distribution $F_n(x)$ and theoretical distribution F(x), using Kolmogorov-Smirnov's limiting distribution.

For this purpose:

1. The order statistics $\{x(i)\}$ are determined from the set $\{x_i\}$ by sorting $\{x_i\}$ into a nondecreasing sequence.

2. The empirical cumulative distribution function $F_n(x)$ is computed. This is the following step-function:

$$F_n(x) = \begin{cases} 0 & x < x_{(1)} \\ k/n & x_{(k)} \le x < x_{(k+1)}; \; k=1,\ldots,n-1 \\ 1 & x_{(n)} \le x \end{cases}$$

3. The maximum deviation D_n in absolute value between the empirical and theoretical distribution is computed:

$$D_n = \max_{-\infty<x<\infty} \left| F_n(x) - F(x) \right|$$

Since $F_n(x)$ and F(x) are nondecreasing functions, the result is:

$$D_n = \max_{1\le k\le n} \left| F_n\left[x_{(k)}\right] \quad -F\left[x_{(k)}\right] \right|$$

D_n is a random variable, and L(z) is the limiting cumulative distribution function of $n^{1/2}\, D_n$:

$$\lim_{n\to\infty} \text{Prob} \left\{ n^{1/2} \; D_n \; < \; z \right\} = L(z)$$

4. Finally, the values are computed for:

$$z = n^{1/2} \; D_n$$

and the probability of being greater than or equal to the computed value of $n^{1/2}\; D_n$ is computed:

$$P = 1 - L(z)$$

Generally, theoretical distribution functions are to be included by the user, as specified in the program. However, four functions are evaluated in KLMO, as follows:

$$\int_{-\infty}^{x} dF(t) = F(x) \qquad (1)$$

is evaluated at the points of the set $\{X_{(i)}\}$, where F(x) is one of the following:

- The normal pdf with mean u and variance s^2

- The exponential pdf with mean u and variance s^2

- The Cauchy pdf with median u, and first quartile s - u

- The uniform pdf with endpoints u and s

Any user-written pdf should evaluate equation (1) above, using the parameters u and s at his convenience. Instructions given in the program KLMO should be followed.

Lilliefors (1967) notes that critical values determined by this test are not correct when one or more parameters are estimated from the sample. The user should refer to his article for notes on approximations that may be considered if such estimates are used.

Programming Considerations:

It is doubtful that the user will wish to perform this test using double-precision accuracy. However, if one wishes to communicate with KLMO in a double-precision program, he might declare

```
XX FLOAT BINARY (53)
X  FLOAT BINARY
```

Before calling KLMO, the user might do the following:

```
DO I = 1 TO N, .
X(I)  = XX(I), .
END, .
```

After exiting from KLMO, the user might do the the following:

```
DO I = 1 TO N, .
XX(I) = X(I), .
END, .
```

(Note that subroutine SMIR has the double-precision option.)

- Subroutine KLM2

```
KLM2..
/**********************************************************************/
/*                                                                    */
/*      TESTS THE DIFFERENCE BETWEEN TWO SAMPLE DISTRIBUTION          */
/*      FUNCTIONS USING THE KOLMOGOROV-SMIRNOV TEST.                  */
/*                                                                    */
/**********************************************************************/
   PROCEDURE(X,Y,N,M,Z,PROB),.
   DECLARE
        (X(*),Y(*),TEMP,XM1,XN1,Z,PROB,D) FLOAT BINARY,
        (I,J,K,L,M,N) FIXED BINARY,
        EPROR EXTERNAL CHARACTER (1),.
   ERROR='0',.
   IF N LT 100 OR M LT 100                /* M OR N IS LESS THAN 100     */
   THEN IF N=0 OR M=0                     /* SET ERROR INDICATOR         */
        THEN DO,.
             ERROF='4',.
             GO TO S60,.
             END,.
        ELSE ERROR='3',.
        DO I=1 TO N-1,.                   /* SORT X INTO                 */
             DO J=I+1 TO N,.              /* ASCENDING SEQUENCE          */
             IF X(I) GT X(J)
             THEN DO,.
                  TEMP =X(I),.
                  X(I) =X(J),.
                  X(J) =TEMP,.
                  END,.
             END,.
        END,.
        DO I=1 TO M-1,.                   /* SORT Y INTO                 */
             DO J=I+1 TO M,.              /* ASCENDING SEQUENCE          */
             IF Y(I) GT Y(J)
             THEN DO,.
                  TEMP =Y(I),.
                  Y(I) =Y(J),.
                  Y(J) =TEMP,.
                  END,.
             END,.
        END,.
   XN1  =1/FLOAT(N),.                     /* CALC. D=ABS(FN-GM)          */
   XM1  =1/FLOAT(M),.                     /* OVER THE SPECTRUM OF X & Y  */
   D,I,J,K,L =0,.
S10..
   IF Y(J+1) GT X(I+1)
   THEN DO,.
        K=1,.
S20..
        I=I+1,.
        IF N LE I
        THEN DO,.
             L=1,.
             GO TO S30,.
             END,.
        ELSE IF X(I) GE X(I+1)
             THEN GO TO S20,.
             ELSE
S30..
                  IF K = 0
                  THEN
S40..
                       DO,.
                       J=J+1,.
                       IF J LT M
                       THEN IF Y(J+1) LE Y(J)
                            THEN GO TO S40,.
                            ELSE GO TO S50,.
                       ELSE DO,.
                            L=1,.
                            GO TO S50,.
                            END,.
                       END,.
                  ELSE GO TO S50,.
        END,.
   ELSE IF X(I+1) = Y(J+1)
        THEN DO,.
             K=0,.
             GO TO S20,.
             END,.
        ELSE GO TO S40,.
                                          /* CHOOSE THE MAXIMUM          */
                                          /* DIFFERENCE, D               */
S50..
   D    =MAX(D,ABS(FLOAT(I)*XN1-FLOAT(J)*XM1)),.
   IF L=0
   THEN GO TO S10,.
   ELSE DO,.
                                          /* CALCULATE THE STATISTIC Z   */
                                          /* AND Z'S PROBABILITY         */
        Z    =D*SQRT((FLOAT(N)*FLOAT(M))/(FLOAT(N)+FLOAT(M))),.
        CALL SMIR (Z,PROB),.
        END,.
S60..
   RETURN,.
   END,.                                  /* END OF PROCEDURE KLM2       */
```

Purpose:

KLM2 tests the difference between two sample distribution functions using the Kolmogorov-Smirnov test.

Usage:

CALL KLM2 (X, Y, N, M, Z, PROB);

X(N) - BINARY FLOAT
Given vector containing N independent observations.

Y(M) - BINARY FLOAT
Given vector containing M independent observations.

N - BINARY FIXED
Given number of observations in X.

M - BINARY FIXED
Given number of observations in Y.

Z - BINARY FLOAT
Resultant variable containing the greatest value with respect to the spectrum of X and Y of

$$\sqrt{\frac{MN}{M+N}}\ \left(\left|F_N(x)-G_M(y)\right|\right)$$

where F_N is the empirical distribution function of the set (x) and $G_M(y)$ is the empirical distribution function of the set (y).

PROB - BINARY FLOAT
Resultant variable containing the probability of the statistic being greater than or equal to Z if the hypothesis that X and Y are from the same PDF is true. For example, PROB=0.05 implies that one can reject the null hypothesis that the sets X and Y are from the same density with 5% probability of being incorrect. PROB=1-SMIR (Z).

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=3 - number of observations N, or number of observations M, less than 100.
ERROR=4 - number of observations N, or number of observations M, equal to zero.

See the mathematical background for this subroutine and for subroutine SMIR, concerning asymptotic formulae.

Subroutines and function subroutines required:
SMIR

Method:

For references see:

W. Feller, "On the Kolmogorov-Smirnov limit theorems for empirical distributions", Annals of Math. Stat., 19, pp. 177-189.

N. Smirnov, "Table for estimating the goodness of fit of empirical distributions", Annals of Math. Stat., 19, pp. 279-281.

B.V. Gnedenko, The Theory of Probability. Chelsea Publishing Co. New York, 1962, pp. 384-401.

Mathematical Background:

Given a sample of n i.i.d. (independent and indentically distributed) random variables X, and a sample of m i.i.d. random variables Y, this subroutine tests the difference between the two empirical distribution functions $F_n(x)$ and $G_m(y)$ using Kolmogorov-Smirnov's limiting distribution. For this purpose:

1. The sets X and Y are sorted into the ordered sets $\{X_{(i)}\}$ and $\{Y_{(i)}\}$, which are nondecreasing sequences.
2. The empirical cumulative distribution functions $F_n(x)$ for the set X, and $G_m(y)$ for the set Y, are computed. For example,

$$F_n(x) = \begin{cases} 0 & x < x_{(1)} \\ k/n & x_{(k)} \le x < x_{(k+1)};\ k=1,\dots,n-1 \\ 1 & x_{(n)} \le x \end{cases}$$

3. The maximum difference in absolute value between the two sample distribution functions is computed:

$$D_{m,n} = \max_{x,\, y} \left| F_n(x) - G_m(y) \right|$$

The statistic $\sqrt{\frac{mn}{m+n}}\, D_{m,n}$ is a random variable with limiting cumulative distribution function L(z), which is described under subroutine SMIR in this manual. That is,

$$\lim_{m, n \to \infty, \infty} \text{Prob} \left\{ \sqrt{\frac{mn}{m+n}}\, D_{m,n} < z \right\} = L(z)$$

4. Finally, the probability (asymptotic) of the statistic $\sqrt{\frac{mn}{m+n}}\, D_{m,n}$ being not less than its computed value, under the assumption of equality of the two theoretical distribution functions from which X and Y were taken, is computed:

$$P = 1 - L(z)$$

Programming Considerations:

It is doubtful that the user will wish to perform this test using double-precision accuracy. However, if one wishes to communicate with KLM2 in a double-precision program, he might declare

```
(XX, YY)  FLOAT BINARY (53)
(Y,X)     FLOAT BINARY
```

giving X and XX, Y and YY the same dimensions.

Before calling KLM2, he might do the following:

```
DO I=1 TO N,.        DO J=1 TO M,.
X(I) =XX(I),.        Y(J) =YY(J),.
END,.                END,.
```

Immediately after exiting from KLM2, he might do the following:

```
DO I=1 TO N,.        DO J=1 TO M,.
XX(I)=X(I),.         YY(J)=Y(J),.
END,.                END,.
```

• Subroutine SMIR

```
SMIR..
/**********************************************************************/
/*                                                                    */
/*      COMPUTES VALUES OF THE LIMITING DISTRIBUTION FUNCTION FOR THE*/
/*      KOLMOGOROV-SMIRNOV STATISTIC.                                 */
/*                                                                    */
/**********************************************************************/
  PROCEDURE (X,Y),.
  DECLARE
      (X,Y,Q1,Q2,Q4,Q8) FLOAT BINARY,./*SINGLE PRECISION          /*S*/
 /*   (X,Y,Q1,Q2,Q4,Q8) FLOAT BINARY (53),./*DOUBLE PRECISION     /*D*/
  IF X LT 1.0
  THEN IF X LE .27                      /* X LESS THAN .27-SET Y      */
      THEN Y    =0.0,.
                                        /* CALCULATE L(X)             */
                                        /* IN RANGE (.27,1)           */
      ELSE DO,.
          Q1    =EXP(-1.233701E0/X**2),.      /* SINGLE PREC.   /*S*/
 /*       Q1    =EXP(-1.2337005013617OE0/X**2),.
                                              /* DOUBLE PREC.   /*D*/
          Q2    =Q1*Q1,.
          Q4    =Q2*Q2,.
          Q8    =Q4*Q4,.
          IF Q8-1.0E-25 GE 0
          THEN Y     =(2.506628E0/X)*Q1*(1.0E0+Q8*(1.0E0+Q8*Q8)),.
                                              /* SINGLE PREC.   /*S*/
 /*       THEN Y     =(2.5066282746310O1E0/X)*Q1*(1.0E0+Q8*
                      (1.0E0+Q8*Q8)),.        /* DOUBLE PREC.   /*D*/
          ELSE Y     =(2.506628E0/X)*Q1,.     /* SINGLE PREC.   /*S*/
 /*       ELSE Y     =(2.506628274631001E0/X)*Q1,.
                                              /* DOUBLE PREC.   /*D*/
          END,.
  ELSE IF X LT 3.1
                                        /* CALCULATE L(X)             */
      THEN DO,.                         /* IN RANGE (1,3.1)           */
          Q1    =EXP(-2.0E0*X*X),.
          Q2    =Q1*Q1,.
          Q4    =Q2*Q2,.
          Q8    =Q4*Q4,.
          Y     =1.0EC-2.0E0*(Q1-Q4+Q8*(Q1-Q8)),.
          END,.
      ELSE Y    =1.0,.                  /* X > OR = 3.1--SET Y        */
  RETURN,.
  END,.                                 /* END OF PROCEDURE SMIR      */
```

Purpose:

SMIR computes values of the limiting distribution function for the Kolmogorov-Smirnov statistic.

Usage:

CALL SMIR (X, Y);

X - BINARY FLOAT [(53)]
Given variable containing the argument of the Smirnov function.

Y - BINARY FLOAT [(53)]
Resultant variable containing the Smirnov function value.

Remarks:

Accuracy tests were made referring to the table given in the reference below.

Two arguments, X=.62, and X=1.87, gave results that differ from the Smirnov tables by 2.9 and 1.9 in the 5th decimal place. All other results showed smaller errors, and error specifications are given in the accuracy tables in this manual. In double-precision mode, these same arguments resulted in differences from tabled values by 3 and 2 in the 5th decimal place. It is noted in Lindgren (reference below) that for high-significance levels (say, .01 and .05) asymptotic formulas give values that are too high (by 1.5% when N=80). That is, at high-significance levels, the hypothesis of no difference will be rejected too seldom using asymptotic formulas.

Method:

For references see:

E. T. Whittaker and G. N. Watson, A Course of Modern Analysis, Cambridge University Press, Cambridge, England, 1952, 462-476.

W. Feller, "On the Kolmogorov-Smirnov limit theorems for empiral distributions", Annals of Math. Stat. 19, pp. 177-189.

N. Smirnov, "Table for estimating the goodness of fit of empirical distributions", Annals of Math. Stat. 19, pp. 279-281.

V. W. Lindgren, Statistical Theory, The Macmillan Company, New York, 1962.

Mathematical Background:

This subroutine computes the values of Kolmogorov-Smirnov's limiting distribution for a given argument x.

$$L(x) = \begin{cases} 0 & x \le 0 \\ 1-2\sum_{k=1}^{\infty}(-1)^{k-1}\exp(-2k^2x^2) & x > 0 \end{cases} \qquad (1)$$

L(x) is the limit (Kolmogorov) of the cumulative distribution function of $\sqrt{n}\,D_n$, and of (Smirnov) $[mn/(m+n)]^{1/2}\,D_{m,n}$ where:

D_n is the maximum, over all x, of the difference $|F_n(x) - F(x)|$ between the sample distribution function $F_n(x)$ and the continuous theoretical distribution function F(x), and

$D_{m,n}$ is the maximum, over all x, of the difference between the two sample distribution functions $F_m(x)$ and $G_n(x)$, from two independent samples of sizes m and n.

When x is very small, the series (1) converges slowly, but, using Jacobi's theta-functions $\theta_2(u,t)$ and $\theta_4(u,t)$:

$$\theta_2(u,t) = 2\sum_{k=0}^{\infty} \exp\,[i\pi(k+1/2)^2t]\cos[(2k+1)u]$$

$$\theta_4(u,t) = 1-2\sum_{k=0}^{\infty}(-1)^{k-1}\exp(i\pi k^2t)\cos(2ku)$$

and using the Jacobi imaginary transformation

$$\theta_4(0,t) = (-it)^{-1/2}\,\theta_2(0,\ -1/t)$$

it follows that:

$$L(x) = \theta_4(0,\ 2ix^2/\pi)$$

$$= (\sqrt{2\pi}/x) \sum_{k=1}^{\infty} \exp\left[-(2k-1)^2\,\pi^2/8x^2\right]$$

which converges quickly when x is small. The computation here uses, with errors $E_i(x)$, i=1,2:

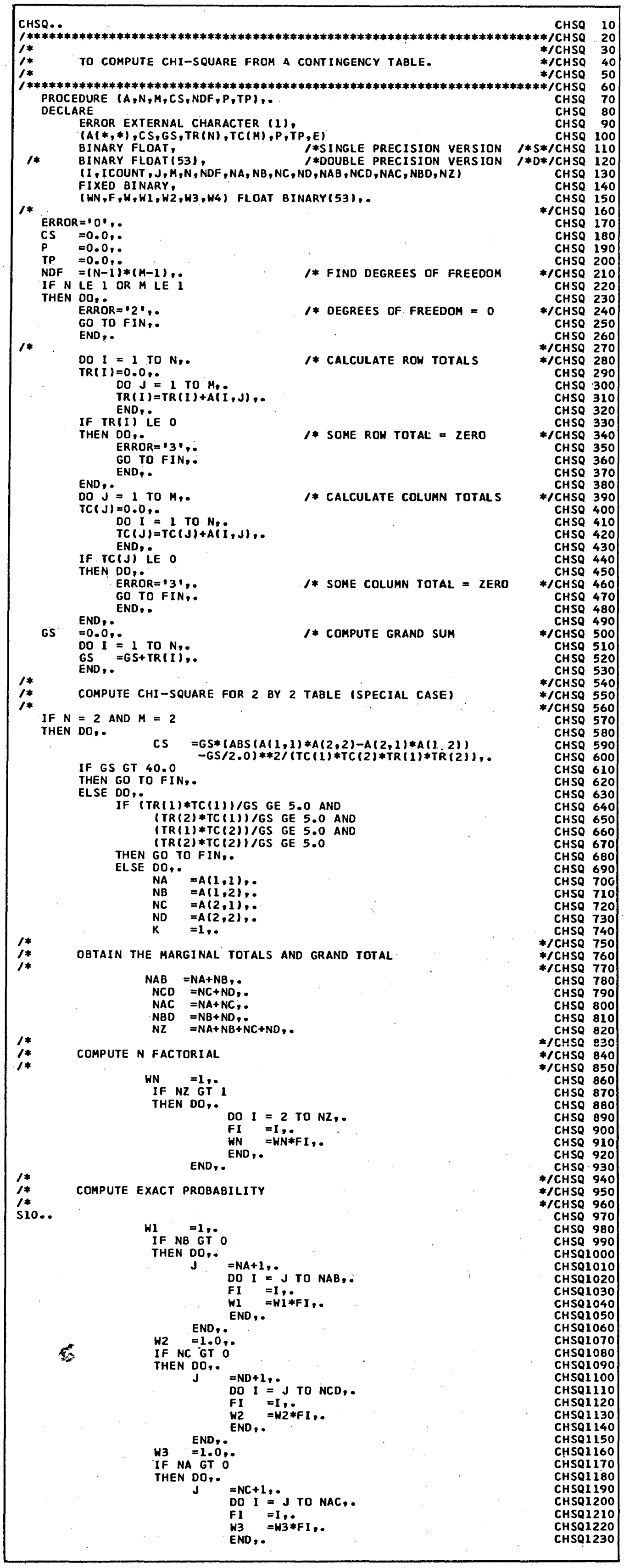
$$L(x) = \begin{cases} 0 & x \leq 0.27 \\ (\sqrt{2\pi}/x) \sum_{k=1}^{3} \exp\left[-(2k-1)^2\pi^2/8x^2\right] + E_1(x); & 0.27 < x < 1.0 \\ 1 - 2\sum_{k=1}^{4} (-1)^{k-1} \exp(-2k^2x^2) + E_2(x) & 1.0 \leq x < 3.1 \\ 1 & 3.1 \leq x < \infty \end{cases}$$

where:

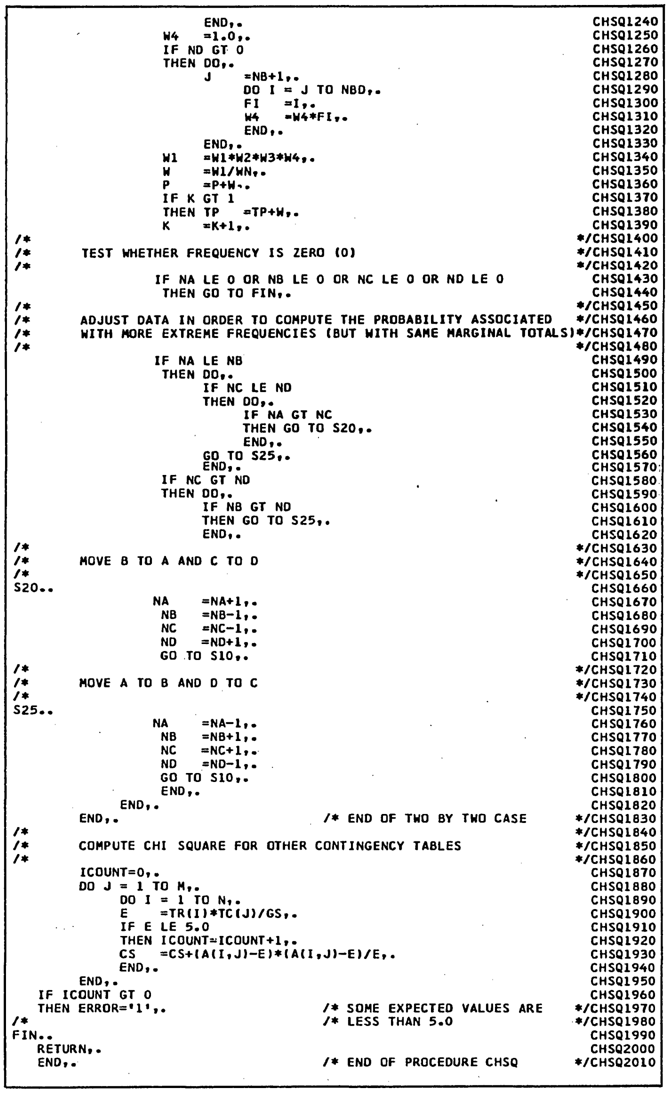
$$E_1(x) \leq 6\,(10^{-15}) \text{ when } x < 1$$

$$E_2(x) < 10^{-20} \qquad \text{when } x \geq 1$$

- <u>Subroutine</u> CHSQ



```
CHSQ..
/**********************************************************************/
/*                                                                    */
/*      TO COMPUTE CHI-SQUARE FROM A CONTINGENCY TABLE.               */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,N,M,CS,NDF,P,TP),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (A(*,*),CS,GS,TR(N),TC(M),P,TP,E)
        BINARY FLOAT,                 /*SINGLE PRECISION VERSION   /*S*/
 /*     BINARY FLOAT(53),             /*DOUBLE PRECISION VERSION   /*D*/
        (I,ICOUNT,J,M,N,NDF,NA,NB,NC,ND,NAB,NCD,NAC,NBD,NZ)
        FIXED BINARY,
        (WN,F,W,W1,W2,W3,W4) FLOAT BINARY(53),.
/*                                                                    */
   ERROR='0',.
   CS   =0.0,.
   P    =0.0,.
   TP   =0.0,.
   NDF  =(N-1)*(M-1),.                /* FIND DEGREES OF FREEDOM      */
   IF N LE 1 OR M LE 1
   THEN DO,.
        ERROR='2',.                   /* DEGREES OF FREEDOM = 0       */
        GO TO FIN,.
        END,.
/*                                                                    */
        DO I = 1 TO N,.               /* CALCULATE ROW TOTALS         */
        TR(I)=0.0,.
             DO J = 1 TO M,.
             TR(I)=TR(I)+A(I,J),.
             END,.
        IF TR(I) LE 0
        THEN DO,.                     /* SOME ROW TOTAL = ZERO        */
             ERROR='3',.
             GO TO FIN,.
             END,.
        END,.
        DO J = 1 TO M,.               /* CALCULATE COLUMN TOTALS      */
        TC(J)=0.0,.
             DO I = 1 TO N,.
             TC(J)=TC(J)+A(I,J),.
             END,.
        IF TC(J) LE 0
        THEN DO,.
             ERROR='3',.              /* SOME COLUMN TOTAL = ZERO     */
             GO TO FIN,.
             END,.
        END,.
   GS   =0.0,.                        /* COMPUTE GRAND SUM            */
        DO I = 1 TO N,.
        GS   =GS+TR(I),.
        END,.
/*                                                                    */
/*      COMPUTE CHI-SQUARE FOR 2 BY 2 TABLE (SPECIAL CASE)            */
/*                                                                    */
   IF N = 2 AND M = 2
   THEN DO,.
                 CS   =GS*(ABS(A(1,1)*A(2,2)-A(2,1)*A(1.2))
                       -GS/2.0)**2/(TC(1)*TC(2)*TR(1)*TR(2)),.
        IF GS GT 40.0
        THEN GO TO FIN,.
        ELSE DO,.
             IF (TR(1)*TC(1))/GS GE 5.0 AND
                  (TR(2)*TC(1))/GS GE 5.0 AND
                  (TR(1)*TC(2))/GS GE 5.0 AND
                  (TR(2)*TC(2))/GS GE 5.0
             THEN GO TO FIN,.
             ELSE DO,.
                  NA   =A(1,1),.
                  NB   =A(1,2),.
                  NC   =A(2,1),.
                  ND   =A(2,2),.
                  K    =1,.
/*                                                                    */
/*      OBTAIN THE MARGINAL TOTALS AND GRAND TOTAL                    */
/*                                                                    */
                 NAB  =NA+NB,.
                  NCD  =NC+ND,.
                  NAC  =NA+NC,.
                  NBD  =NB+ND,.
                  NZ   =NA+NB+NC+ND,.
/*                                                                    */
/*      COMPUTE N FACTORIAL                                           */
/*                                                                    */
                 WN    =1,.
                  IF NZ GT 1
                  THEN DO,.
                            DO I = 2 TO NZ,.
                            FI   =I,.
                            WN   =WN*FI,.
                            END,.
                       END,.
/*                                                                    */
/*      COMPUTE EXACT PROBABILITY                                     */
/*                                                                    */
S10..
                 W1    =1,.
                  IF NB GT 0
                  THEN DO,.
                       J    =NA+1,.
                            DO I = J TO NAB,.
                            FI   =I,.
                            W1   =W1*FI,.
                            END,.
                       END,.
                  W2   =1.0,.
                  IF NC GT 0
                  THEN DO,.
                       J    =ND+1,.
                            DO I = J TO NCD,.
                            FI   =I,.
                            W2   =W2*FI,.
                            END,.
                       END,.
                  W3   =1.0,.
                  IF NA GT 0
                  THEN DO,.
                       J    =NC+1,.
                            DO I = J TO NAC,.
                            FI   =I,.
                            W3   =W3*FI,.
                            END,.
```



```
                    END,.                                                CHSQ1240
                W4    =1.0,.                                             CHSQ1250
                IF ND GT 0                                               CHSQ1260
                THEN DO,.                                                CHSQ1270
                    J     =NB+1,.                                        CHSQ1280
                        DO I = J TO NBD,.                                CHSQ1290
                        FI    =I,.                                       CHSQ1300
                        W4    =W4*FI,.                                   CHSQ1310
                        END,.                                            CHSQ1320
                    END,.                                                CHSQ1330
                W1    =W1*W2*W3*W4,.                                     CHSQ1340
                W     =W1/WN,.                                           CHSQ1350
                P     =P+W-.                                             CHSQ1360
                IF K GT 1                                                CHSQ1370
                THEN TP   =TP+W,.                                        CHSQ1380
                K     =K+1,.                                             CHSQ1390
/*                                                                     */CHSQ1400
/*      TEST WHETHER FREQUENCY IS ZERO (0)                             */CHSQ1410
/*                                                                     */CHSQ1420
               IF NA LE 0 OR NB LE 0 OR NC LE 0 OR ND LE 0               CHSQ1430
                THEN GO TO FIN,.                                         CHSQ1440
/*                                                                     */CHSQ1450
/*      ADJUST DATA IN ORDER TO COMPUTE THE PROBABILITY ASSOCIATED     */CHSQ1460
/*      WITH MORE EXTREME FREQUENCIES (BUT WITH SAME MARGINAL TOTALS)*/CHSQ1470
/*                                                                     */CHSQ1480
               IF NA LE NB                                               CHSQ1490
                THEN DO,.                                                CHSQ1500
                    IF NC LE ND                                          CHSQ1510
                    THEN DO,.                                            CHSQ1520
                        IF NA GT NC                                      CHSQ1530
                        THEN GO TO S20,.                                 CHSQ1540
                        END,.                                            CHSQ1550
                    GO TO S25,.                                          CHSQ1560
                    END,.                                                CHSQ1570
                IF NC GT ND                                              CHSQ1580
                THEN DO,.                                                CHSQ1590
                    IF NB GT ND                                          CHSQ1600
                    THEN GO TO S25,.                                     CHSQ1610
                    END,.                                                CHSQ1620
/*                                                                     */CHSQ1630
/*      MOVE B TO A AND C TO D                                         */CHSQ1640
/*                                                                     */CHSQ1650
S20..                                                                    CHSQ1660
               NA    =NA+1,.                                             CHSQ1670
                NB   =NB-1,.                                             CHSQ1680
                NC   =NC-1,.                                             CHSQ1690
                ND   =ND+1,.                                             CHSQ1700
                GO TO S10,.                                              CHSQ1710
/*                                                                     */CHSQ1720
/*      MOVE A TO B AND D TO C                                         */CHSQ1730
/*                                                                     */CHSQ1740
S25..                                                                    CHSQ1750
               NA    =NA-1,.                                             CHSQ1760
                NB   =NB+1,.                                             CHSQ1770
                NC   =NC+1,.                                             CHSQ1780
                ND   =ND-1,.                                             CHSQ1790
                GO TO S10,.                                              CHSQ1800
                END,.                                                    CHSQ1810
            END,.                                                        CHSQ1820
        END,.                             /* END OF TWO BY TWO CASE    */CHSQ1830
/*                                                                     */CHSQ1840
/*      COMPUTE CHI SQUARE FOR OTHER CONTINGENCY TABLES                */CHSQ1850
/*                                                                     */CHSQ1860
        ICOUNT=0,.                                                       CHSQ1870
        DO J = 1 TO M,.                                                  CHSQ1880
            DO I = 1 TO N,.                                              CHSQ1890
            E    =TR(I)*TC(J)/GS,.                                       CHSQ1900
            IF E LE 5.0                                                  CHSQ1910
            THEN ICOUNT=ICOUNT+1,.                                       CHSQ1920
            CS   =CS+(A(I,J)-E)*(A(I,J)-E)/E,.                           CHSQ1930
            END,.                                                        CHSQ1940
        END,.                                                            CHSQ1950
    IF ICOUNT GT 0                                                       CHSQ1960
    THEN ERROR='1',.                      /* SOME EXPECTED VALUES ARE  */CHSQ1970
/*                                        /* LESS THAN 5.0             */CHSQ1980
FIN..                                                                    CHSQ1990
    RETURN,.                                                             CHSQ2000
    END,.                                 /* END OF PROCEDURE CHSQ     */CHSQ2010
```

Purpose:

CHSQ computes chi-square from a contingency table.

Usage:

CALL CHSQ (A, N, M, CS, NDF, P, TP);

A(N,M) - BINARY FLOAT [(53)]
Given matrix containing contingency table of integer values.

N - BINARY FIXED
Given number of rows in matrix A.

M - BINARY FIXED
Given number of columns in matrix A.

CS - BINARY FLOAT [(53)]
Resultant chi-square.

NDF - BINARY FIXED
Resultant number of degrees of freedom.

P - BINARY FLOAT [(53)]
Resultant exact probability for a 2x2 contingency table. If the contingency table is not 2x2, the value of P will be zero (P=0).

TP - BINARY FLOAT [(53)]
Resultant variable containing the probability by the Tocher-modification method for a 2x2 contingency table. If the contingency table is not 2x2, the value of TP will be set to zero (TP=0).

Remarks:

P, CS, and TP above are computed only when the contingency table is 2x2, the total of the frequencies is less than or equal to 40, and the expected frequency in any cell is less than five.

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - some expected values less than 5.0.
ERROR=2 - degrees of freedom equal to zero.
ERROR=3 - some row total or column total less than or equal to zero.

Method:

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

Mathematical Background:

When the observations are classified by two characteristics (two-way classification), the chi-square test may be used to test the hypothesis that the two characteristics are independent--namely, that the distribution of one characteristic is the same regardless of the other characteristic. Two-way-classification tables of this type are frequently called contingency tables, and different formulas are used to compute chi-square for the following two types of contingency tables:

1. For 2 x 2 table:

$$\text{a. } \chi^2 = \frac{N\left(\left| AD-BC \right| - \frac{N}{2}\right)^2}{(A+B)\ (C+D)\ (A+C)\ (B+D)} \qquad (1)$$

where A, B, C, and D stand for frequencies in a 2 x 2 table as shown below, and N=A+B+C+D.

	Yes	No
Male	A	B
Female	C	D

b. If N ≤ 40 and the expected frequency in any cell is 5, the Fisher exact probability is computed.

The exact probability of observing a particular set of frequencies in a 2 x 2 table, when the marginal totals are regarded as fixed, is given by the formula:

$$p = \frac{(A+B)!\ (C+D)!\ (A+C)!\ (B+D)!}{N!\ A!\ B!\ C!\ D!} \qquad (2)$$

However, more extreme distributions of frequencies could occur with the same marginal totals.

To find the Fisher exact probability, we add the probability of obtaining the existing distribution of frequencies to the probabilities of obtaining all the more extreme distributions of frequencies.

The more extreme distributions of frequencies are determined by systematically subtracting one from the smallest frequency in the table, while keeping the marginal totals fixed. This iterative process continues until the smallest cell has a zero value. This is the most extreme case.

$$p_F = p_a + p_b + p_c + \ldots .$$

For example:

Observed data — More extreme outcomes with same marginal totals

table a

2	6	8
4	2	6
6	8	14

table b

1	7	8
5	1	6
6	8	14

table c

0	8	8
6	0	6
6	8	14

$$p_a = \frac{8!\ 6!\ 6!\ 8!}{14!\ 2!\ 6!\ 4!\ 2!} = 20/143 = .13986$$

$$p_b = \frac{8!\ 6!\ 6!\ 8!}{14!\ 1!\ 7!\ 5!\ 1!} = 16/1001 = .01598$$

$$p_c = \frac{8!\ 6!\ 6!\ 8!}{14!\ 0!\ 8!\ 6!\ 0!} = 1/3003 = .00033$$

The probability associated with the occurrence of values as extreme or more extreme than those observed (table a) is given by adding the three probabilities

$$.13986 + .01598 + .00033 = .15617$$

Thus, $p_F = .15617$ is the Fisher exact probability.

Tocher's modification determines the probability of all cases more extreme than the observed one, and not including the observed one.

$$p_T = p_b + p_c + \ldots . \qquad (3)$$

That is,

$$p_T = p_F - p_a \qquad (4)$$

For the example in tables a, b, and c:

$p_T = .01598 + .00033 = .01631$ using equation (3)

$p_T = .15617 - .13986 = .01631$ using equation (4)

The probability (p_T) provided by Tocher's modification to the Fisher exact test is for a one-tailed test of H_0. For a two-tailed test, the p_T yielded must be doubled.

2. For other contingency tables:

$$\chi^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \qquad (5)$$

where:

A_{ij} = frequency in the cell i, j

$$E_{ij} = \frac{T_i T_j}{N} \qquad (6)$$

$$T_i = \sum_{j=1}^{n} A_{ij} \qquad i = 1, 2, \ldots, n \text{ (row totals)} \qquad (7)$$

$$T_j = \sum_{i=1}^{n} A_{ij} \quad j = 1, 2, \ldots, m \text{ (column totals)} \tag{8}$$

$$N = \sum_{i=1}^{n} T_i \quad \text{(grand total)} \tag{9}$$

The degrees of freedom:

$$\text{d. f.} = (n - 1)\ (m - 1) \tag{10}$$

- Subroutine KRNK

```
KRNK..
/***********************************************************************/
/*                                                                     */
/*       TO TEST CORRELATION BETWEEN TWO VARIABLES BY MEANS OF THE     */
/*       KENDALL RANK CORRELATION COEFFICIENT.                         */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (A,B,R1,R2,N,TAU,SD,Z,NR),.
   DECLAFE
       (A(*),B(*),R1(*),R2(*),TAU,SD,Z,RSAVE,SAVER,S,TA,TB,FN1,FN)
       FLOAT BINARY,
       (I,ISORT,J,KT,N,NR)
       BINARY FIXED,
       ERROR EXTERNAL CHARACTER (1),.
/*                                                                     */
   ERROR='0',.                         /* INITIALIZATION               */
       DO I=1 TO N,.
       R1(I)     =0,.
       R2(I)     =0,.
       END,.
   TAU  =0.0,.
   SD   =0.0,.
   Z    =0.0,.
   IF N LE 1                           /* NUMBER OF OBSERVATIONS LESS  */
   THEN DO,.                           /* THAN OR EQUAL TO ONE.        */
       ERROR='1',.
       GO TO FIN,.
       END,.
   FN   =N,.
   FN1  =N*(N-1),.
   IF NR= 1                            /* DETERMINE IF DATA IS RANKED  */
   THEN DO,.
           DO I = 1 TO N,.
           R1(I)=A(I),.                /* MOVE RANKED DATA TO R1 R2    */
           R2(I)=B(I),.
           END,.
       END,.
   ELSE DO,.
/*                                                                     */
/*       RANK DATA IN A AND B VECTORS AND ASSIGN TIED OBSERVATIONS     */
/*       AVERAGE OF TIED RANKS.                                        */
/*                                                                     */
       CALL RANK (A,R1,N),.
       CALL RANK (B,R2,N),.
       END,.
S10..
   ISORT=0,.
/*                                                                     */
/*       SORT RANK VECTORS R1 AND R2 IN SEQUENCE OF VARIABLE A         */
/*                                                                     */
       DO I = 2 TO N,.
       IF R1(I) LT R1(I-1)
       THEN DO,.
           ISORT=ISORT+1,.
           RSAVE=R1(I),.
           R1(I)=R1(I-1),.
           R1(I-1)=RSAVE,.
           SAVER=R2(I),.
           R2(I)=R2(I-1),.
           R2(I-1)=SAVER,.
           END,.
       END,.
   IF ISORT NE 0
   THEN GO TO S1C,.
/*                                                                     */
/*       COMPUTE S ON VARIABLE B. STARTING WITH THE FIRST RANK, ADD 1  */
/*       TO S FOR EACH LARGER RANK TO ITS RIGHT AND SUBTRACT 1 FOR     */
/*       EACH SMALLER RANK. REPEAT FOR ALL RANKS.                      */
/*                                                                     */
   S    =0,.
       DO I = 1 TO N-1,.
           DO J = I+1 TO N,.
           IF R2(J) GT R2(I)
           THEN S     =S+1.0,.
           ELSE IF R2(J) LT R2(I)
           THEN S     =S-1.0,.
           END,.
       END,.
/*                                                                     */
/*       COMPUTE TIED SCORE INDEX FOP BOTH VARIABLES                   */
/*                                                                     */
   KT   =2,.
   CALL TIE (F1,N,KT,TA),.
   IF ERROR='2'
   THEN
S20..
       DO,.                            /* ALL RANKS FOR ONE VARIABLE   */
       EFROF='3',.                     /* APE EQUAL                    */
       GO TO FIN,.
       END,.
   CALL TIE (R2,N,KT,TB),.
   IF ERROF='2'
   THEN GO TO S20,.
   IF TA= 0.0 AND TB = 0.0             /* COMPUTE TAU                  */
   THEN TAU  =S/(0.5*FN1),.
   ELSE TAU  =S/((SORT(C.5*FN1-TA))*(SQRT(0.5*FN1-T8))),.
/*                                                                     */
/*       COMPUTE STANDARD DEVIATION AND Z VALUE IF N IS 10 OR GREATER */
/*                                                                     */
   IF N GE 10
   THEN DO,.
       SD   =(SQRT((2.0*(FN+FN+5))/(9.0*FN1))),.
       Z    =TAU/SD,.
       END,.
   ELSE ERROR='2',.                    /* SAMPLE SIZE LESS THAN 10     */
FIN..
   RETURN,.
   END,.                               /*END OF PROCEDURE KRNK         */
```

Purpose:

KRNK measures the correlation between two variables by means of the Kendall rank correlation coefficient.

Usage:

CALL KRNK (A, B, R1, R2, N, TAU, SD, Z, NR);

A(N) - BINARY FLOAT
Given vector containing observations for the first variable.

B(N) - BINARY FLOAT
Given vector containing observations for the second variable.

R1(N) - BINARY FLOAT
Resultant vector containing rank of the data in vector A.

R2(N) - BINARY FLOAT
Resultant vector containing rank of the data in vector B.

N - BINARY FIXED
Given number of observations.

TAU - BINARY FLOAT
Resultant variable containing the Kendall rank correlation coefficient.

SD - BINARY FLOAT
Resultant variable containing standard deviation.

Z - BINARY FLOAT
Resultant variable containing statistic to be used to measure the significance of TAU in terms of normal distribution.

NR - BINARY FIXED
Given code containing the following:
0 - for raw data in vectors A and B.
1 - for the rank of data in vectors A and B.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations less than or equal to one.

ERROR=2 - sample size less then 10. If this condition exists, R1 and R2 will contain invalid values; SD and Z will be set to zero.

ERROR=3 - all ranks for one variable are equal.

Subroutines and function subroutines required:
RANK
TIE

Method:

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

Mathematical Background:

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:

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

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

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

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

(1) if T_a and T_b are zero,

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

where:

n = number of ranks
S = total score calculated for ranks in variable B by selecting each rank in turn, adding 1 for each larger rank to its right, subtracting 1 for each smaller rank to its right.

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

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

The standard deviation is calculated:

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

The statistic used to measure the significance of τ is:

$$z = \frac{\tau}{s}$$

- <u>Subroutine</u> QTST

```
QTST..
/***********************************************************************/
/*                                                                     */
/*      TO TEST WHETHER THREE OR MORE MATCHED GROUPS OF DICHOTOMOUS    */
/*      DATA DIFFER SIGNIFICANTLY BY THE COCHRAN Q-TEST.               */
/*                                                                     */
/***********************************************************************/
   PROCEDURE (A,N,M,Q,NDF),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (A(*,*),TR(N),TC(M),Q,RSQ,CSQ,GD,FM)
        BINARY FLOAT,
        (I,J,M,N,NDF)
        BINARY FIXED,.
/*                                                                     */
   ERROR='0',.
   IF M LT 3 OR N LE 1                  /* NUMBER OF CASES IN EACH     */
   THEN DO,.                            /* GROUP IS LESS THAN 3 OR     */
        ERROR='1',.                     /* THE NUMBER OF OBSERVATIONS  */
        GO TO FIN,.                     /* IS LESS THAN OR EQUAL TO    */
        END,.                           /* ONE.                        */
   FM   =M,.
/*                                                                     */
/*      COMPUTE SUM OF SQUARES OF ROW AND COLUMN TOTALS RSQ AND CSQ,   */
/*      AND GRAND TOTAL OF ALL ELEMENTS.                               */
/*                                                                     */
        DO I = 1 TO N,.
        TR(I)=0.0,.
             DO J = 1 TO M,.
             TR(I)=TR(I)+A(I,J),.
             END,.
        END,.
        DO J = 1 TO M,.                 /* CALCULATE COLUMN SUMS       */
        TC(J)=0.0,.
             DO I = 1 TO N,.
             TC(J)=TC(J)+A(I,J),.
             END,.
        END,.
   Q    =0.0,.
   NDF  =0.0,.
   GD   =0.0,.
   RSQ  =0.0,.
   CSQ  =0.0,.
        DO I = 1 TO N,.
        GD   =GD+TR(I),.                /* GRAND TOTAL                 */
        RSQ  =RSQ+TR(I)*TR(I),.         /* SUM OF ROW TOTAL SQUARED    */
        END,.
        DO J = 1 TO M,.
        CSQ  =CSQ+TC(J)*TC(J),.         /* SUM OF COLUMN TOTAL SQUARED */
        END,.
   Q    =FM*GD-RSQ,.
   IF Q LT 1                            /* TEST FOR Q NEAR ZERO        */
   THEN DO,.
        ERROR='2',.
        GO TO FIN,.
        END,.
/*                                                                     */
/*    COMPUTE COCHRAN Q TEST VALUE.                                    */
/*                                                                     */
   Q    =(FM-1.0)*(FM*CSQ-GD*GD)/(FM*GD-RSQ),.
   NDF  =M-1,.                          /* FIND DEGREES OF FREEDOM     */
FIN..
   RETURN,.
   END,.                                /*END OF PROCEDURE QTST        */
```

Purpose:

QTST uses the Cochran Q-test to determine whether three or more matched groups of dichotomous data differ significantly.

Usage:

CALL QTST (A, N, M, Q, NDF);

A(N,M) - BINARY FLOAT
Given matrix of dichotomous data. Data elements must be either 0 or 1.

N - BINARY FIXED
Given number of sets in each group.

M - BINARY FIXED
Given number of groups.

Q - BINARY FLOAT
Resultant Cochran Q statistic.

NDF - BINARY FIXED
Resultant number of degrees of freedom.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of groups, M, is less than three and/or the number of sets, N, is less than or equal to one.

ERROR=2 - all values of matrix A are equal.

Method:

Described in S. Siegel, <u>Nonparameteric Statistics for the Behavioral Sciences</u>, McGraw-Hill, New York, 1956, chapter 7.

Mathematical Background:

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:

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

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

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

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

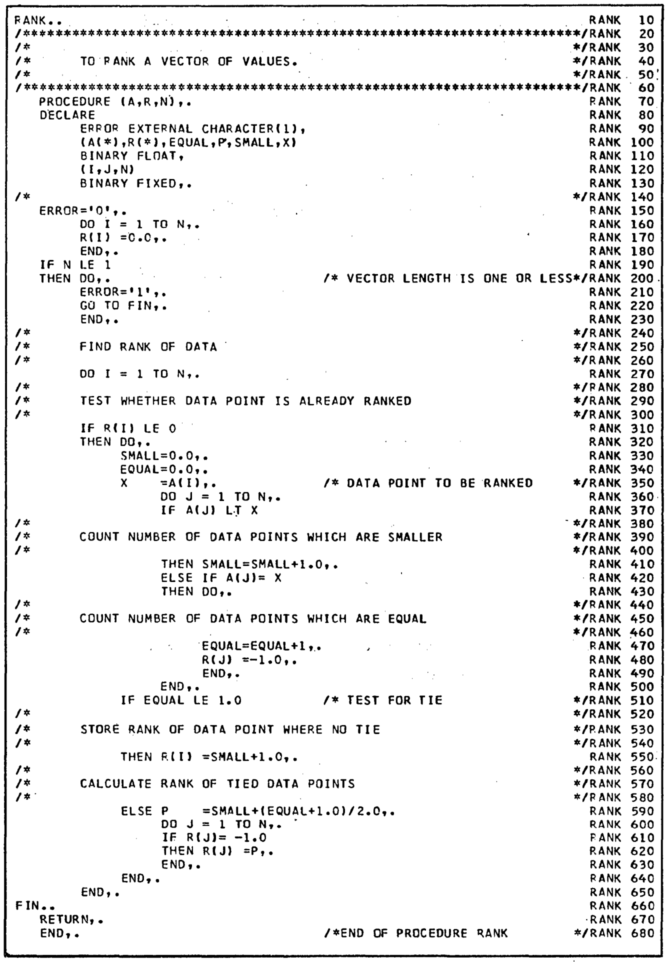
The Cochran Q statistic is computed:

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

The degrees of freedom are:

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

- Subroutine RANK



```
RANK..
/**********************************************************************/
/*                                                                    */
/*      TO RANK A VECTOR OF VALUES.                                   */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,R,N),.
   DECLARE
        ERROR EXTERNAL CHARACTER(1),
        (A(*),R(*),EQUAL,P,SMALL,X)
        BINARY FLOAT,
        (I,J,N)
        BINARY FIXED,.
/*                                                                    */
   ERROR='0',.
        DO I = 1 TO N,.
        R(I) =0.0,.
        END,.
   IF N LE 1
   THEN DO,.                          /* VECTOR LENGTH IS ONE OR LESS*/
        ERROR='1',.
        GO TO FIN,.
        END,.
/*                                                                    */
/*      FIND RANK OF DATA                                             */
/*                                                                    */
        DO I = 1 TO N,.
/*                                                                    */
/*      TEST WHETHER DATA POINT IS ALREADY RANKED                     */
/*                                                                    */
        IF R(I) LE 0
        THEN DO,.
             SMALL=0.0,.
             EQUAL=0.0,.
             X    =A(I),.             /* DATA POINT TO BE RANKED      */
                  DO J = 1 TO N,.
                  IF A(J) LT X
/*                                                                    */
/*      COUNT NUMBER OF DATA POINTS WHICH ARE SMALLER                 */
/*                                                                    */
                  THEN SMALL=SMALL+1.0,.
                  ELSE IF A(J)= X
                  THEN DO,.
/*                                                                    */
/*      COUNT NUMBER OF DATA POINTS WHICH ARE EQUAL                   */
/*                                                                    */
                       EQUAL=EQUAL+1,.
                       R(J) =-1.0,.
                       END,.
                  END,.
             IF EQUAL LE 1.0          /* TEST FOR TIE                 */
/*                                                                    */
/*      STORE RANK OF DATA POINT WHERE NO TIE                         */
/*                                                                    */
             THEN R(I) =SMALL+1.0,.
/*                                                                    */
/*      CALCULATE RANK OF TIED DATA POINTS                            */
/*                                                                    */
             ELSE P    =SMALL+(EQUAL+1.0)/2.0,.
                  DO J = 1 TO N,.
                  IF R(J)= -1.0
                  THEN R(J) =P,.
                  END,.
             END,.
        END,.
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE RANK         */
```

Purpose:

RANK ranks a vector of data.

Usage:

CALL RANK (A, R, N);

A(N) - BINARY FLOAT
Given vector containing data to be ranked.

R(N) - BINARY FLOAT
Resultant vector containing the ranks of the data in A. Smallest value is ranked 1; largest is ranked N. Ties are assigned the average of the tied ranks.

N - BINARY FIXED
Given number of values.

Remarks:

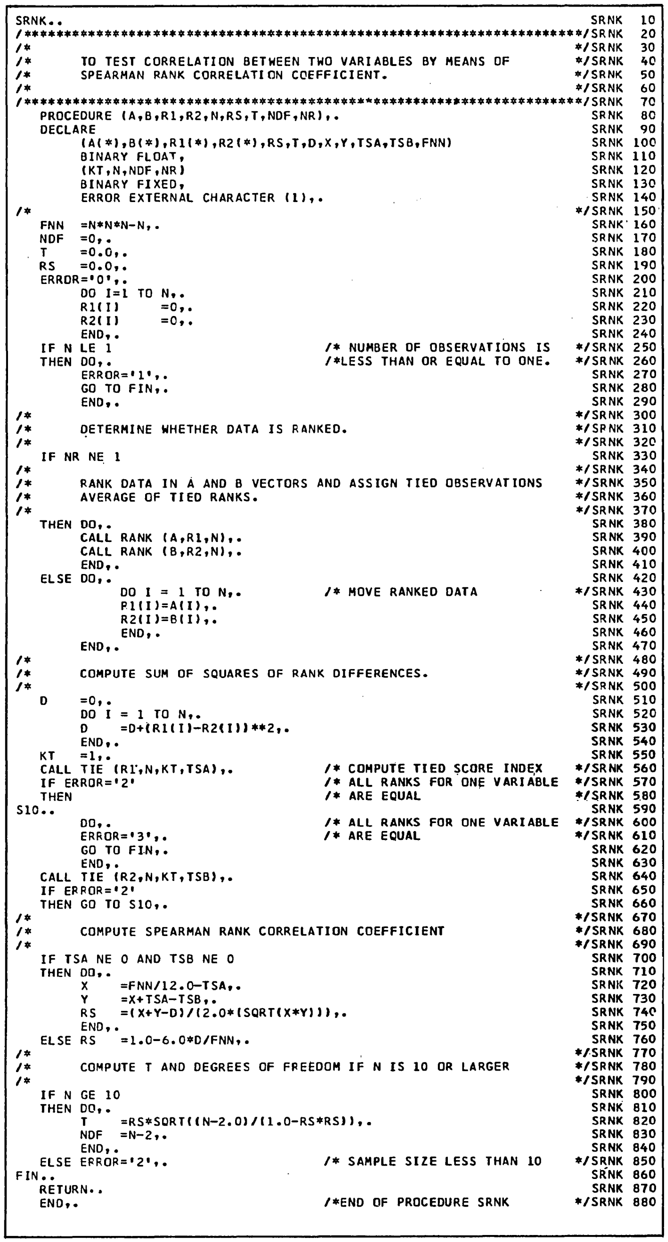
If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR=1 - vector length one or less.

Method:

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

- Subroutine SRNK



```
SRNK..
/**********************************************************************/
/*                                                                    */
/*      TO TEST CORRELATION BETWEEN TWO VARIABLES BY MEANS OF         */
/*      SPEARMAN RANK CORRELATION COEFFICIENT.                        */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,B,R1,R2,N,RS,T,NDF,NR),.
   DECLARE
        (A(*),B(*),R1(*),R2(*),RS,T,D,X,Y,TSA,TSB,FNN)
        BINARY FLOAT,
        (KT,N,NDF,NR)
        BINARY FIXED,
        ERROR EXTERNAL CHARACTER (1),.
/*                                                                    */
   FNN  =N*N*N-N,.
   NDF  =0,.
   T    =0.0,.
   RS   =0.0,.
   ERROR='0',.
        DO I=1 TO N,.
        R1(I)     =0,.
        R2(I)     =0,.
        END,.
   IF N LE 1                           /* NUMBER OF OBSERVATIONS IS   */
   THEN DO,.                           /*LESS THAN OR EQUAL TO ONE.   */
        ERROR='1',.
        GO TO FIN,.
        END,.
/*                                                                    */
/*      DETERMINE WHETHER DATA IS RANKED.                             */
/*                                                                    */
   IF NR NE 1
/*                                                                    */
/*      RANK DATA IN A AND B VECTORS AND ASSIGN TIED OBSERVATIONS     */
/*      AVERAGE OF TIED RANKS.                                        */
/*                                                                    */
   THEN DO,.
        CALL RANK (A,R1,N),.
        CALL RANK (B,R2,N),.
        END,.
   ELSE DO,.
              DO I = 1 TO N,.          /* MOVE RANKED DATA            */
              P1(I)=A(I),.
              R2(I)=B(I),.
              END,.
        END,.
/*                                                                    */
/*      COMPUTE SUM OF SQUARES OF RANK DIFFERENCES.                   */
/*                                                                    */
   D    =0,.
        DO I = 1 TO N,.
        D    =D+(R1(I)-R2(I))**2,.
        END,.
   KT   =1,.
   CALL TIE (R1,N,KT,TSA),.            /* COMPUTE TIED SCORE INDEX    */
   IF ERROR='2'                        /* ALL RANKS FOR ONE VARIABLE  */
   THEN                                /* ARE EQUAL                   */
S10..
        DO,.                           /* ALL RANKS FOR ONE VARIABLE  */
        ERROR='3',.                    /* ARE EQUAL                   */
        GO TO FIN,.
        END,.
   CALL TIE (R2,N,KT,TSB),.
   IF ERROR='2'
   THEN GO TO S10,.
/*                                                                    */
/*      COMPUTE SPEARMAN RANK CORRELATION COEFFICIENT                 */
/*                                                                    */
   IF TSA NE 0 AND TSB NE 0
   THEN DO,.
        X    =FNN/12.0-TSA,.
        Y    =X+TSA-TSB,.
        RS   =(X+Y-D)/(2.0*(SQRT(X*Y))),.
        END,.
   ELSE RS   =1.0-6.0*D/FNN,.
/*                                                                    */
/*      COMPUTE T AND DEGREES OF FREEDOM IF N IS 10 OR LARGER         */
/*                                                                    */
   IF N GE 10
   THEN DO,.
        T    =RS*SQRT((N-2.0)/(1.0-RS*RS)),.
        NDF  =N-2,.
        END,.
   ELSE ERROR='2',.                    /* SAMPLE SIZE LESS THAN 10    */
FIN..
   RETURN..
   END,.                               /*END OF PROCEDURE SRNK        */
```

Purpose:

SRNK tests the correlation between two variables by means of the Spearman rank correlation coefficient.

Usage:

CALL SRNK (A, B, R1, R2, N, RS, T, NDF, NR);

A(N) - BINARY FLOAT
Given vector containing the observations for the first variable.

B(N) - BINARY FLOAT
Given vector containing the observations for the second variable.

R1(N) - BINARY FLOAT
Resultant vector containing rank of the data in vector A.

R2(N) - BINARY FLOAT
Resultant vector containing rank of the data in vector B.

N - BINARY FIXED
Given number of observations.

RS - BINARY FLOAT
Resultant variable containing the Spearman rank correlation coefficients.

T - BINARY FLOAT
Resultant variable containing the measure to be used to test the significance of RS.

NDF - BINARY FIXED
Resultant variable containing the number of degrees of freedom.

NR - BINARY FIXED
Given code containing the following:
0 - for raw data in vectors A and B.
1 - for the rank of data in vectors A and B.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of observations less than or equal to one. If this condition exists, R1 and R2 will contain invalid values.

ERROR=2 - sample size less than 10. (T and NDF are not computed if this condition is detected.)

ERROR=3 - All ranks for one variable are equal.

Procedures and function procedures required:
RANK
TIE

Method:

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

Mathematical Background:

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:

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

where:

A_i = first ranked vector

B_i = second ranked vector

n = number of ranks

A correction factor for ties is obtained:

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

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

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

(1) if T_a and T_b are zero

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

(2) if T_a and/or T_b are not zero

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

where:

$$X = \frac{N^3 - N}{12} - T_a \qquad (5)$$

$$Y = \frac{N^3 - N}{12} - T_b \qquad (6)$$

The significance of r_s can be measured by the statistic:

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

The degrees of freedom are:

$$d.\ f. = N-2 \qquad (8)$$

- Subroutine TIE

```
TIE..
/**********************************************************************/
/*                                                                    */
/*      TO CALCULATE CORRELATION FACTOR DUE TO TIES.                 */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (R,N,KT,T),.
   DECLAPE
        (R(*),T,X,Y,CT)
        BINARY FLOAT,  .
        ERROR EXTERNAL CHARACTER(1),
        (I,IND,KT,N)
        BINARY FIXED,.
/*                                                                    */
   ERROR='0',.
   IF N LE 1
   THEN DO,.                          /* VECTOR LENGTH IS ONE OR LESS*/
        ERROR='1',.
        GO TO FIN,.
        END,.
   T    =0.0,.                        /* INITIALIZATION               */
   Y    =0.0,.
S10..
   X    =N+1,.
   IND  =0,.
        DO I = 1 TO N,.               /* FIND NEXT LARGEST RANK       */
        IF R(I) GT Y AND R(I) LT X
        THEN DO,.
             X    =R(I),.
             IND  =IND+1,.
             END,.
        END,.
/*                                                                    */
/*      IF ALL RANKS HAVE BEEN TESTED RETURN                          */
/*                                                                    */
   IF IND NE 0
   THEN DO,.
        Y    =X,.
        CT   =0.0,.
             DO I = 1 TO N,.          /* COUNT TIES                   */
             IF R(I)= X
             THEN CT   =CT+1.0,.
             END,.
        IF CT NE 0.0
        THEN DO,.
             IF KT= 1
             THEN T    =T+(CT*CT*CT-CT)/12.0,.
             ELSE T    =T+CT*(CT-1.0)/2.0,.
             END,.
        GO TO S10,.
        END,.
FIN..
   IF CT=N                            /* ALL RANKS FOR ONE VARIABLE   */
   THEN ERROR='2',.                   /* ARE EQUAL                    */
   RETURN,.
   END,.                              /*END OF PROCEDURE TIE          */
```

Purpose:

TIE calculates correction factor due to ties.

Usage:

CALL TIE (R, N, KT, T);

R(N) - BINARY FLOAT
Given vector of ranks containing values from 1 to N.

N - BINARY FIXED
Given number of ranked values.

KT - BINARY FIXED
Given code for calculation of correction factor
1 - solve equation 1
2 - solve equation 2

T - BINARY FLOAT
Resultant variable containing correction factor
Equation 1 T=SUM(CT**3-CT)/12
Equation 2 T=SUM(CT*(CT-1)/2)
where CT is the number of observations tied for a given rank.

Remarks:

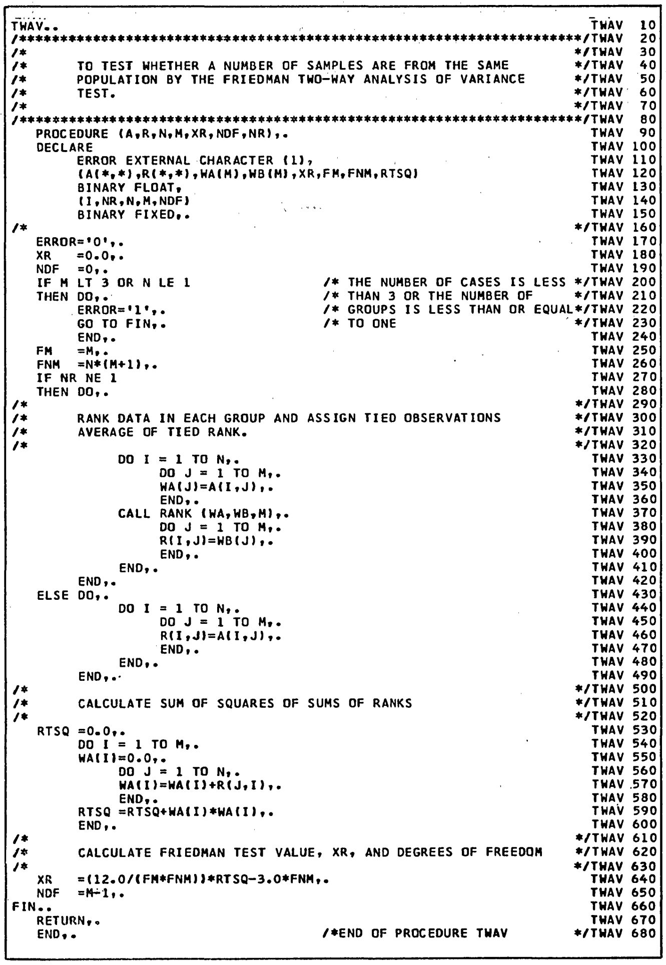
If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - vector length one or less.
ERROR=2 - all ranks of one variable are equal.

Method:

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

- Subroutine TWAV



```
TWAV..
/**********************************************************************/
/*                                                                    */
/*     TO TEST WHETHER A NUMBER OF SAMPLES ARE FROM THE SAME          */
/*     POPULATION BY THE FRIEDMAN TWO-WAY ANALYSIS OF VARIANCE        */
/*     TEST.                                                          */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (A,R,N,M,XR,NDF,NR),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (A(*,*),R(*,*),WA(M),WB(M),XR,FM,FNM,RTSQ)
        BINARY FLOAT,
        (I,NR,N,M,NDF)
        BINARY FIXED,.
/*                                                                    */
   ERROR='0',.
   XR   =0.0,.
   NDF  =0,.
   IF M LT 3 OR N LE 1                /* THE NUMBER OF CASES IS LESS */
   THEN DO,.                          /* THAN 3 OR THE NUMBER OF     */
        ERROR='1',.                   /* GROUPS IS LESS THAN OR EQUAL*/
        GO TO FIN,.                   /* TO ONE                      */
        END,.
   FM   =M,.
   FNM  =N*(M+1),.
   IF NR NE 1
   THEN DO,.
/*                                                                    */
/*     RANK DATA IN EACH GROUP AND ASSIGN TIED OBSERVATIONS           */
/*     AVERAGE OF TIED RANK.                                          */
/*                                                                    */
             DO I = 1 TO N,.
                  DO J = 1 TO M,.
                  WA(J)=A(I,J),.
                  END,.
             CALL RANK (WA,WB,M),.
                  DO J = 1 TO M,.
                  R(I,J)=WB(J),.
                  END,.
             END,.
        END,.
   ELSE DO,.
             DO I = 1 TO N,.
                  DO J = 1 TO M,.
                  R(I,J)=A(I,J),.
                  END,.
             END,.
        END,.
/*                                                                    */
/*     CALCULATE SUM OF SQUARES OF SUMS OF RANKS                      */
/*                                                                    */
   RTSQ =0.0,.
        DO I = 1 TO M,.
        WA(I)=0.0,.
             DO J = 1 TO N,.
             WA(I)=WA(I)+R(J,I),.
             END,.
        RTSQ =RTSQ+WA(I)*WA(I),.
        END,.
/*                                                                    */
/*     CALCULATE FRIEDMAN TEST VALUE, XR, AND DEGREES OF FREEDOM      */
/*                                                                    */
   XR   =(12.0/(FM*FNM))*RTSQ-3.0*FNM,.
   NDF  =M-1,.
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE TWAV        */
```

Purpose:

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

Usage:

CALL TWAV (A, R, N, M, XR, NDF, NR);

A(N,M) - BINARY FLOAT
Given matrix of original data.
R(N,M) - BINARY FLOAT
Resultant matrix of the ranks of the data.
N - BINARY FIXED
Given number of groups.
M - BINARY FIXED
Given number of cases in each group.
XR - BINARY FLOAT
Resultant Friedman statistic.
NDF - BINARY FIXED
Resultant number of degrees of freedom.
NR - BINARY FIXED
Given code:
0 for raw data in A;
1 for ranks of the data in A.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitutes the possible error condition that may be detected:

ERROR=1 - number of groups less than or equal to one, or number of cases less than three.

Subroutines and function subroutines required:

RANK

Method:

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

Mathematical Background:

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

The sum of ranks is calculated:

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

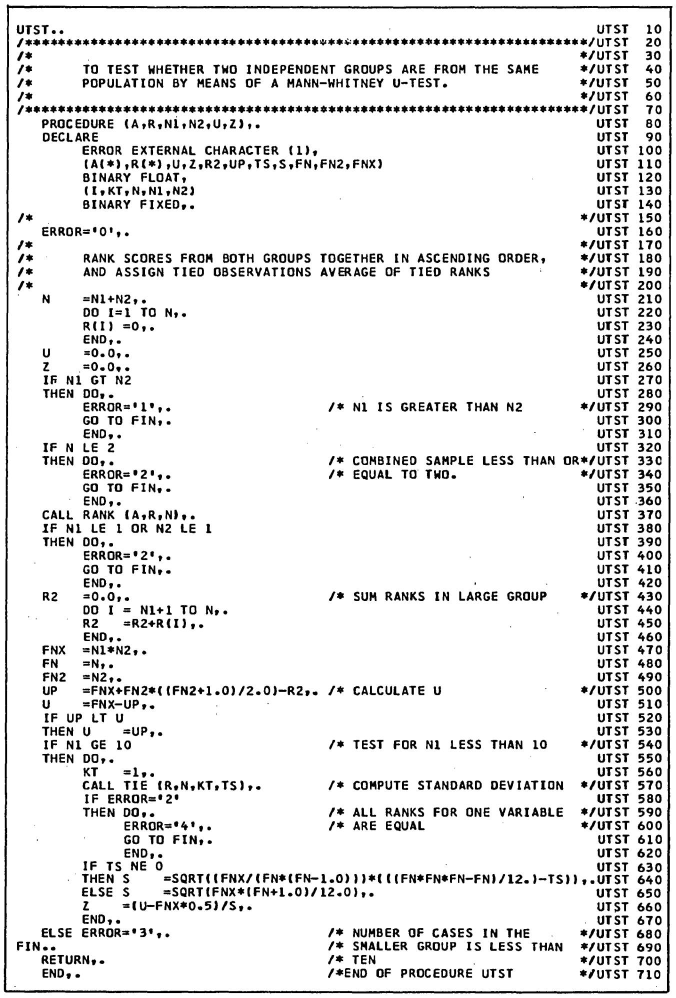
Friedman's statistic is then computed:

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

The degrees of freedom are:

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

® Subroutine UTST



```
UTST..
/*********************************************************************/
/*                                                                   */
/*       TO TEST WHETHER TWO INDEPENDENT GROUPS ARE FROM THE SAME    */
/*       POPULATION BY MEANS OF A MANN-WHITNEY U-TEST.               */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,R,N1,N2,U,Z),.
   DECLARE
         ERROR EXTERNAL CHARACTER (1),
         (A(*),R(*),U,Z,R2,UP,TS,S,FN,FN2,FNX)
         BINARY FLOAT,
         (I,KT,N,N1,N2)
         BINARY FIXED,.
/*                                                                   */
   ERROR='0',.
/*                                                                   */
/*       RANK SCORES FROM BOTH GROUPS TOGETHER IN ASCENDING ORDER,   */
/*       AND ASSIGN TIED OBSERVATIONS AVERAGE OF TIED RANKS          */
/*                                                                   */
   N     =N1+N2,.
         DO I=1 TO N,.
         R(I) =0,.
         END,.
   U     =0.0,.
   Z     =0.0,.
   IF N1 GT N2
   THEN DO,.
         ERROR='1',.                     /* N1 IS GREATER THAN N2      */
         GO TO FIN,.
         END,.
   IF N LE 2
   THEN DO,.                             /* COMBINED SAMPLE LESS THAN OR*/
         ERROR='2',.                     /* EQUAL TO TWO.              */
         GO TO FIN,.
         END,.
   CALL RANK (A,R,N),.
   IF N1 LE 1 OR N2 LE 1
   THEN DO,.
         ERROR='2',.
         GO TO FIN,.
         END,.
   R2    =0.0,.                          /* SUM RANKS IN LARGE GROUP   */
         DO I = N1+1 TO N,.
         R2    =R2+R(I),.
         END,.
   FNX   =N1*N2,.
   FN    =N,.
   FN2   =N2,.
   UP    =FNX+FN2*((FN2+1.0)/2.0)-R2,. /* CALCULATE U                */
   U     =FNX-UP,.
   IF UP LT U
   THEN U     =UP,.
   IF N1 GE 10                           /* TEST FOR N1 LESS THAN 10   */
   THEN DO,.
         KT    =1,.
         CALL TIE (R,N,KT,TS),.          /* COMPUTE STANDARD DEVIATION */
         IF ERROR='2'
         THEN DO,.                       /* ALL RANKS FOR ONE VARIABLE */
               ERROR='4',.               /* ARE EQUAL                  */
               GO TO FIN,.
               END,.
         IF TS NE 0
         THEN S     =SQRT((FNX/(FN*(FN-1.0)))*(((FN*FN*FN-FN)/12.)-TS)),.
         ELSE S     =SQRT(FNX*(FN+1.0)/12.0),.
         Z     =(U-FNX*0.5)/S,.
         END,.
   ELSE ERROR='3',.                      /* NUMBER OF CASES IN THE     */
FIN..                                    /* SMALLER GROUP IS LESS THAN */
   RETURN,.                              /* TEN                        */
   END,.                                 /*END OF PROCEDURE UTST       */
```

Purpose:

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

Usage:

CALL UTST (A, R, N1, N2, U, Z);

A(N) - BINARY FLOAT
Given vector of cases consisting of two independent groups. Smaller group precedes larger group. N = N1 + N2.

R(N) - BINARY FLOAT
Resultant vector of ranks. Smallest value is ranked 1; largest is ranked N. Ties are assigned average of tied ranks.

N1 - BINARY FIXED
Given number of cases in smaller group.

N2 - BINARY FIXED
Given number of cases in larger group.

U - BINARY FLOAT
Resultant statistic used to test homogeneity of the two groups.

Z - BINARY FLOAT
Resultant measure for determining the significance of U in terms of normal distribution (if N1 is less than 10, Z is set to zero).

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - N1 greater than N2.
ERROR=2 - Combined samples less than or equal to two.
ERROR=3 - number of cases in the smaller group is less than 10 (in this case Z is set to zero).
ERROR=4 - all ranks for one variable are equal.

Subroutines and function subroutines required:

RANK
TIE

Method:

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

Mathematical Background:

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 the smaller group preceding the larger group. The scores for both groups are ranked together in ascending order. Tied observations are assigned the average of the tied ranks.

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

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

where:

n_1 = number of cases in smaller group

n_2 = number of cases in larger group

$U = n_1 n_2 - U'$

if U' < U, set U = U'

$$\text{if } U' < U, \text{ set } U = U' \tag{2}$$

A correction factor for ties is obtained:

$$T = \sum \frac{t^3 - t}{12} \tag{3}$$

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

The standard deviation is computed for two cases:

(1) if T = 0

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

(2) if T > 0

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

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

The measure used to determine the significance of U is then calculated:

$$Z = \frac{U - \overline{X}}{S} \tag{6}$$

where $\overline{X}$ = mean = $\frac{N1\ N2}{2}$

Z is set to zero if N1 is less than 10.

- Subroutine WTST

```
WTST..
/*********************************************************************/
/*                                                                   */
/*      TO TEST DEGREE OF ASSOCIATION AMONG A NUMBER OF VARIABLES    */
/*      BY THE KENDALL COEFFICIENT OF CONCORDANCE.                   */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,R,N,M,W,CS,NDF,NR),.
   DECLARE
        ERROR EXTERNAL CHARACTER (1),
        (A(*,*),R(*,*),WA(M),WB(M),W,CS,SM,S,TI,T,FN,FM)
        BINARY FLOAT,
        (I,J,KT,M,N,NDF,NR)
        BINARY FIXED,.
/*                                                                   */
   ERROR='0',.
        DO I=1 TO N,.
             DO J=1 TO M,.
             R(I,J)    =0,.
             END,.
        END,.
   W    =0.0,.
   CS   =0.0,.
   NDF  =0,.
   IF N LT 3 OR M LT 3
   THEN DO,.
        ERROR='1',.                   /* NUMBER OF VARIABLES (N) OR  */
        GO TO FIN,.                   /* NUMBER OF CASES (M) IS LESS */
        END,.                         /* THAN 3                      */
/*                                                                   */
/*      DETERMINE WHETHER DATA IS RANKED. IF IT HAS NOT BEEN DONE    */
/*      RANK DATA FOR ALL VARIABLES ASSIGNING TIED OBSERVATIONS      */
/*      AVERAGE OF TIED RANKS AND COMPUTE CORRECTION FOR TIED SCORES */
/*                                                                   */
   T    =0,.
   KT   =1,.
        DO I = 1 TO N,.
        IF NR NE 1
        THEN DO,.
                  DO J = 1 TO M,.
                  WA(J)=A(I,J),.
                  END,.
             CALL RANK (WA,WB,M),.
             END,.
        ELSE DO,.
                  DO J = 1 TO M,.
                  WB(J)=A(I,J),.
                  END,.
             END,.
        CALL TIE (WB,M,KT,TI),.
        IF ERROR='2'
        THEN DO,.
             ERROR='3',.              /* ALL RANKS FOR ONE VARIABLE  */
             GO TO FIN,.              /* ARE EQUAL                   */
             END,.
        T    =T+TI,.
             DO J = 1 TO M,.
             R(I,J)=WB(J),.
             END,.
        END,.
   FN   =N,.
   FM   =M,.
   SM   =0.0,.
/*                                                                   */
/*      CALCULATE VECTOR SUMS AND COMPUTE MEANS OF SUMS              */
/*                                                                   */
        DO J = 1 TO M,.
        WA(J)=0.C,.
             DO I = 1 TO N,.
             WA(J)=WA(J)+R(I,J),.
             END,.
        SM   =SM+WA(J),.
        END,.
   SM   =SM/FM,.
/*                                                                   */
/*      COMPUTE THE SUM OF SQUARES OF DEVIATION                      */
/*                                                                   */
   S    =0,.
        DO J = 1 TO M,.
        S    =S+(WA(J)-SM)**2,.
        END,.
   W    =S/(((FN*FN)*(FM*FM*FM-FM)/12.0)-FN*T),.
/*                                                                   */
/*      COMPUTE DEGREES OF FREEDOM AND CHI-SQUARE IF M IS OVER 7     */
/*                                                                   */
   IF M GT 7
   THEN DO,.
        CS   =FN*(FM-1.0)*W,.
        NDF  =M-1,.
        END,.
   ELSE ERROR='2',.                   /* NUMBER OF CASES (M) IS LESS */
/*                                    /* THAN OR EQUAL TO 7          */
FIN..
   RETURN,.
   END,.                              /*END OF PROCEDURE WTST        */
```

Purpose:

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

Usage:

CALL WTST (A, R, N, M, W, CS, NDF, NR);

A(N, M) - BINARY FLOAT
Given matrix of original data.

R(N, M) - BINARY FLOAT
Resultant matrix, N by M, of the ranks of the data. Smallest value is ranked 1; largest is ranked M. Ties are assigned average of tied ranks. The data is ranked by rows.

N - BINARY FIXED
Given number of variables.

M - BINARY FIXED
Given number of cases.

W - BINARY FLOAT
Resultant variable containing Kendall coefficient of concordance.

CS - BINARY FLOAT
Resultant variable containing the value of chi-square.

NDF - BINARY FIXED
Resultant variable containing number of degrees of freedom.

NR - BINARY FIXED
Given code containing the following:
0 for raw data in A.
1 for the rank of data in A.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR = 1 - number of variables, N, or number of cases, M, less than three.
ERROR = 2 - number of cases, M, less than or equal to seven (CS and NDF are set to zero.)
ERROR = 3 - all ranks for one variable are equal.

Subroutines and function subroutines are required:

RANK
TIE

Method:

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

Mathematical Background:

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

A correction factor for ties is obtained:

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

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

Sums of ranks are calculated:

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

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

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

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

The sum of squares of deviations is derived:

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

The Kendall coefficient of concordance is then computed:

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

For m larger than 7, chi-square is:

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

The degrees of freedom are:

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

Subroutine HTES

```
HTES..
/*********************************************************************/
/*                                                                   */
/*      TO CALCULATE THE KRUSKAL-WALLIS H-STATISTIC FROM THE RANKS   */
/*      CF OBSERVATIONS WHICH ARE OBTAINED FROM THREE OR MORE INDE-  */
/*      PENDENT SAMPLES.                                             */
/*                                                                   */
/*********************************************************************/
   PROCEDURE (A,R,M,NS,H),.
   DECLARE
        (A(*),R(*),H,S,SUMR,T,XK,XN)
        BINARY FLOAT,
        (M(*),I,J,K,L,N,NS)
        BINARY FIXED,
        ERROR EXTERNAL CHARACTER (1),.
/*                                                                   */
        ERROR='0',.                     /* INITIALIZATION            */
   H    =0.0,.
   IF NS LT 3
   THEN ERROR='1',.                     /* SET ERROR INDICATOR       */
   ELSE DO,.
        N    =0,.
             DO I = 1 TO NS,.           /* CALCULATE TOTAL NUMBER OF */
             IF M(I) LE 0               /* CASES IN ALL SAMPLES      */
             THEN DO,.
                  ERROR='3',.
                  GO TO S10,.
                  END,.
             N    =N+M(I),.
             END,.
        XN   =N,.
/*                                                                   */
/*      RANK DATA FROM ALL SAMPLES IN ASCENDING ORDER AND ASSIGN     */
/*      TIED OBSERVATIONS AVERAGE OF TIED RANKS                      */
/*                                                                   */
        CALL RANK (A,R,N),.
/*                                                                   */
        S    =0,.
        J    =0,.
             DO I = 1 TO NS,.
             K    =M(I),.
             XK   =K,.
             SUMR =0.C,.
                  DO L = 1 TO K,.       /* SUM RANKS FOR EACH SAMPLE */
                  J    =J+1,.
                  SUMR =SUMR+R(J),.
                  END,.
             S    =S+SUMR*SUMR/XK,.
             END,.
/*                                                                   */
/*      CALCULATE H, UNCORRECTED FOR TIES                            */
/*                                                                   */
        H    =((12.0*S)/(XN*XN+XN))-3.0*(XN+1),.
/*                                                                   */
/*      COMPUTE CORRECTION FACTOR FOR TIES                           */
/*                                                                   */
        K    =1,.
        CALL TIE (R,N,K,T),.
        IF T = C.0 OR ERROR='2'
        THEN GO TO S10,.
        ELSE DO,.
             S    =1.0-((12.0*T)/(XN**3-XN)),.
/*                                                                   */
/*      CORRECT H FOR TIES                                           */
/*                                                                   */
             H    =H/S,.
             END,.
        END,.
S10..
   RETURN,.
   END,.                                /*END OF PROCEDURE HTES      */
```

Purpose:

HTES calculates the Kruskal-Wallis H-statistic from the ranks of observations obtained from three or more independent samples.

Usage:

CALL HTES (A, R, M, NS, H);

A(N) - BINARY FLOAT
Given vector of observed data stored columnwise. In other words, the data from the first sample, second, third, etc. are stored in consecutive locations of vector A. N=M(1)+M(2)+...+M(NS) (that is, the total number of cases)

R(N) - BINARY FLOAT
Resultant vector containing the ranks of data of vector A. The smallest value is ranked one, and the largest is ranked N. Ties are assigned the average of the tied ranks.

M - BINARY FIXED
Given vector of length NS containing the number of cases in each sample.

NS - BINARY FIXED
Given variable containing the number of samples.

H - BINARY FLOAT
Resultant variable containing the value of H-statistic.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - number of samples, NS, less than three. If this condition exists, R will contain invalid values.
ERROR=2 - all ranks for one variable are equal.
ERROR=3 - the number of cases in one of the samples is less than or equal to zero. If this condition exists, R will contain invalid values.

Subroutines and function subroutines required:

TIE
RANK

Method:

Refer to:

The computational procedures are described in S. Siegel, Nonparametric Statistics for the Behavioral Sciences, McGraw Hill, New York, 1956, chapter 8.

Mathematical Background:

From the data in vector A, the ranks are computed by the subroutine RANK and stored in vector R according to ascending values of the cases, with ties assigned the average of the tied ranks. The ranks are summed for each sample, and the H-statistic is calculated from the formula:

$$H = \left[\frac{12}{N(N+1)} \sum_{i=1}^{NS} \frac{SUMR_i^2}{M_i} \right] -3(N+1) \qquad (1)$$

where:

N = total number of cases

$SUMR_i$ = sum of ranks for the i-th sample
M_i = number of cases in the i-th sample
NS = the number of samples

H is corrected for ties, if present, using the value of T obtained from procedure TIE. The correction formula is:

$$H_{corrected} = \frac{H_{uncorrected}}{1 - \frac{12T}{N^3 - N}} \quad (2)$$

where:

$$T = \sum \frac{(t^3 - t)}{12}, \text{ summed over all samples}$$

t = number of tied observations in a group

H is approximately distributed as χ^2 with (NS-1) degrees of freedom, if the number of cases in each group is not too small (not less than five).

Distribution Functions

- Subroutine NDTR

```
NDTR..
/**********************************************************************/
/*                                                                    */
/*      COMPUTES Y=P(X)=THE PROBABILITY THAT THE RANDOM VARIABLE U,   */
/*      DISTRIBUTED NORMALLY (0,1) IS LESS THAN OR EQUAL TO X.  F(X),*/
/*      THE ORDINATE OF THE NORMAL DENSITY AT X, IS ALSO COMPUTED.    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (X,P,D),.
   DECLARE
        (D,T,P,X,AX) FLOAT BINARY,.
   AX   =ABS(X),.                          /* CALC. PROB. P & DENSITY D   */
   T    =1.0E0/(1.0E0+.2316419E0*AX),.
   D    =0.3989423E0*EXP(-X*X/2.0E0),.
   P    =1.0E0-D*T*((((1.330274E0*T-1.821256E0)*T+1.781478E0)*T-
         0.3565638)*T+0.3193815E0),.
   IF X LT 0                               /* X < 0                       */
   THEN P=1.0E0-P,.                        /* COMPLEMENT PROB. P          */
   RETURN,.
   END,.                                   /* END OF PROCEDURE NDTR       */
```

Purpose:

NDTR computes Y=P(x), the probability that the random available X, distributed normally (0, 1), is less than or equal to x. f(x), the ordinate of the normal density at x, is also computed.

Usage:

CALL NDTR (X, P, D);

X - BINARY FLOAT
Given variable containing the scalar for which P(x) is computed.
P - BINARY FLOAT
Resultant variable containing probability.
D - BINARY FLOAT
Resultant variable containing density.

Method:

Refer to:

C. Hastings, Approximations for Digital Computers. Princeton University Press, Princeton, N.J., 1955.
M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions. Dover Publications, Inc., N.Y., equation 26.2.17.

Mathematical Background:

This subroutine computes y = P(x) = Prob ($X \leq x$), where X is a random variable distributed normally with mean zero and variance one.

$$P(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-u^2/2)du$$

The following approximation is used:

$$P(x) = 1 - f(x) \sum_{i=1}^{5} a_i w^i; \; x \geq 0$$

where:

$w = 1/(1 + px)$

$f(x) = \exp(-x^2/2) / \sqrt{2\pi}$

$P = 0.2316419$

$a_1 = 0.3193815$

$a_2 = -0.3565638$

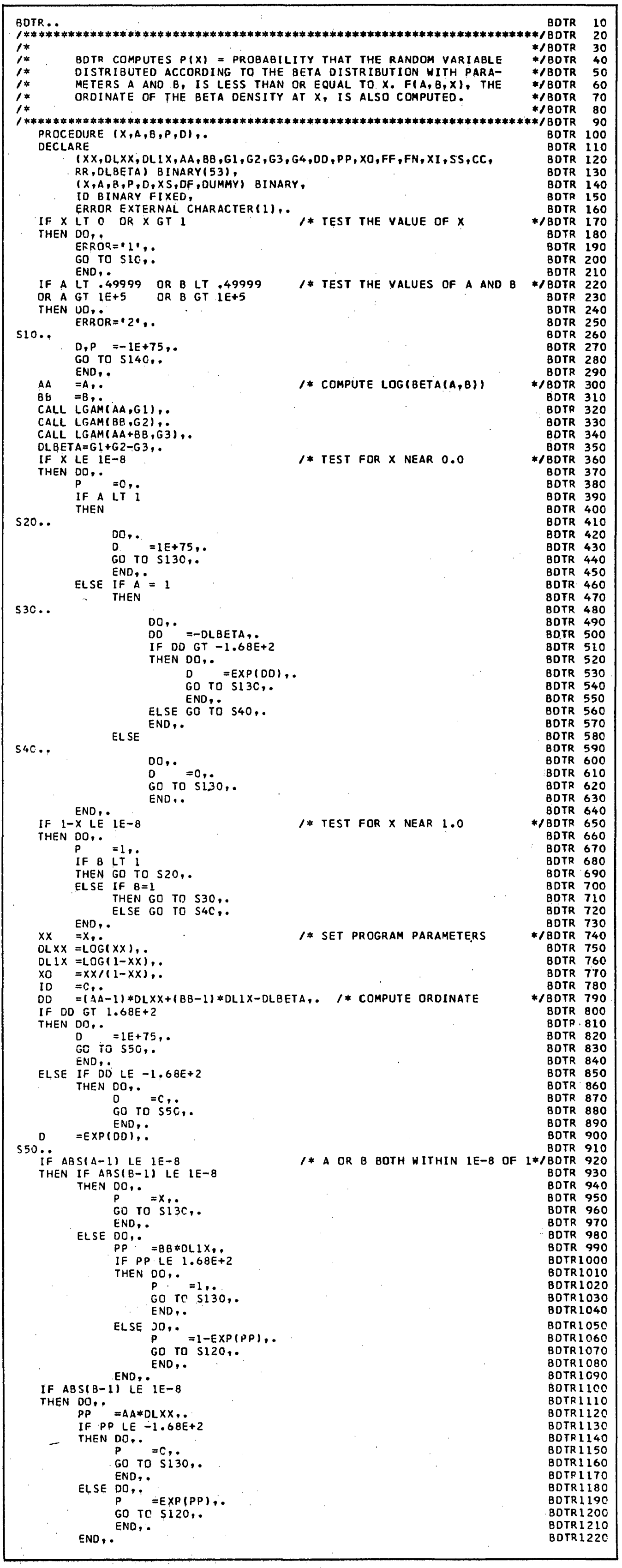
$a_3 = 1.781478$

$a_4 = -1.821256$

$a_5 = 1.330274$

The maximum error is $7(10^{-7})$; f(x) is also presented in output.

- Subroutine BDTR



```
BDTR..
/**********************************************************************/
/*                                                                    */
/*      BDTR COMPUTES P(X) = PROBABILITY THAT THE RANDOM VARIABLE     */
/*      DISTRIBUTED ACCORDING TO THE BETA DISTRIBUTION WITH PARA-     */
/*      METERS A AND B, IS LESS THAN OR EQUAL TO X. F(A,B,X), THE     */
/*      ORDINATE OF THE BETA DENSITY AT X, IS ALSO COMPUTED.          */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (X,A,B,P,D),.
   DECLARE
        (XX,DLXX,DL1X,AA,BB,G1,G2,G3,G4,DD,PP,XO,FF,FN,XI,SS,CC,
        RR,DLBETA) BINARY(53),
        (X,A,B,P,D,XS,DF,DUMMY) BINARY,
        ID BINARY FIXED,
        ERROR EXTERNAL CHARACTER(1),.
   IF X LT 0  OR X GT 1               /* TEST THE VALUE OF X          */
   THEN DO,.
        ERROR='1',.
        GO TO S1C,.
        END,.
   IF A LT .49999  OR B LT .49999     /* TEST THE VALUES OF A AND B   */
   OR A GT 1E+5    OR B GT 1E+5
   THEN DO,.
        ERROR='2',.
S10..
        D,P  =-1E+75,.
        GO TO S140,.
        END,.
   AA   =A,.                          /* COMPUTE LOG(BETA(A,B))       */
   BB   =B,.
   CALL LGAM(AA,G1),.
   CALL LGAM(BB,G2),.
   CALL LGAM(AA+BB,G3),.
   DLBETA=G1+G2-G3,.
   IF X LE 1E-8                       /* TEST FOR X NEAR 0.0          */
   THEN DO,.
        P    =0,.
        IF A LT 1
        THEN
S20..
             DO,.
             D    =1E+75,.
             GO TO S130,.
             END,.
        ELSE IF A = 1
             THEN
S30..
                  DO,.
                  DD   =-DLBETA,.
                  IF DD GT -1.68E+2
                  THEN DO,.
                       D    =EXP(DD),.
                       GO TO S13C,.
                       END,.
                  ELSE GO TO S40,.
                  END,.
             ELSE
S4C..
                  DO,.
                  D    =0,.
                  GO TO S130,.
                  END..
        END,.
   IF 1-X LE 1E-8                     /* TEST FOR X NEAR 1.0          */
   THEN DO,.
        P    =1,.
        IF B LT 1
        THEN GO TO S20,.
        ELSE IF B=1
             THEN GO TO S30,.
             ELSE GO TO S4C,.
        END,.
   XX   =X,.                          /* SET PROGRAM PARAMETERS       */
   DLXX =LOG(XX),.
   DL1X =LOG(1-XX),.
   XO   =XX/(1-XX),.
   ID   =C,.
   DD   =(AA-1)*DLXX+(BB-1)*DL1X-DLBETA,.  /* COMPUTE ORDINATE        */
   IF DD GT 1.68E+2
   THEN DO,.
        D    =1E+75,.
        GO TO S50,.
        END,.
   ELSE IF DD LE -1.68E+2
        THEN DO,.
             D    =C,.
             GO TO S5C,.
             END,.
   D    =EXP(DD),.
S50..
   IF ABS(A-1) LE 1E-8                /* A OR B BOTH WITHIN 1E-8 OF 1*/
   THEN IF ABS(B-1) LE 1E-8
        THEN DO,.
             P    =X,.
             GO TO S13C,.
             END,.
        ELSE DO,.
             PP   =BB*DL1X,,
             IF PP LE 1.68E+2
             THEN DO,.
                  P    =1,.
                  GO TC S130,.
                  END,.
             ELSE DO,.
                  P    =1-EXP(PP),.
                  GO TO S120,.
                  END,.
             END,.
   IF ABS(B-1) LE 1E-8
   THEN DO,.
        PP   =AA*DLXX,.
        IF PP LE -1.68E+2
        THEN DO,.
             P    =C,.
             GO TO S130,.
             END,.
        ELSE DO,,
             P    =EXP(PP),.
             GO TC S120,.
             END,.
        END,.
   IF A GT 1000                        /* TEST FOR A OR B GREATER    */BDTR1230
   THEN DO,.                           /*       THAN 1000            */BDTR1240
        XS   =2*AA/XO,.                                                BDTR1250
        DF   =2*BB,.                                                   BDTR1260
        CALL CDTR(XS,DF,P,DUMMY),.                                     BDTR1270
        P    =1-P,.                                                    BDTR1280
        GO TO S140,.                                                   BDTR1290
        END,.                                                          BDTR1300
   IF B GT 1000                                                        BDTR1310
   THEN DO,.                                                           BDTR1320
        XS   =2*BB*XO,.                                                BDTR1330
        DF   =2*AA,.                                                   BDTR1340
        CALL CDTR(XS,DF,P,DUMMY),.                                     BDTR1350
        GO TO S140,.                                                   BDTR1360
        END,.                                                          BDTR1370
   IF X LE .5                          /* SELECT PARAMETERS FOR CON- */BDTR1380
   THEN IF AA LE 1                     /* TINUED FRACTION COMPUTATION*/BDTR1390
        THEN DO,.                                                      BDTR1400
             RR   =AA+1,.                                              BDTR1410
             GO TO S60,.                                               BDTR1420
             END,.                                                     BDTR1430
        ELSE DO,.                                                      BDTR1440
             RR   =AA,.                                                BDTR1450
S60..                                                                  BDTR1460
             DD   =(RR-1)-(RR+BB-1)*XX*EXP(DLXX/5)+2,.                 BDTR1470
             IF DD LE 0                                                BDTR1480
             THEN GO TO S80,.                                          BDTR1490
             ELSE GO TO S90,.                                          BDTR1500
             END,.                                                     BDTR1510
   IF BB LE 1                                                          BDTR1520
   THEN DO,.                                                           BDTR1530
        RR   =BB+1,.                                                   BDTR1540
        GO TO S70,.                                                    BDTR1550
        END,.                                                          BDTR1560
   RR   =BB,.                                                          BDTR1570
S70..                                                                  BDTR1580
   DD   =(RR-1)-(AA+RR-1)*(1-XX)*EXP(DL1X/5)+2,.                       BDTR1590
   IF DD LE 0                                                          BDTR1600
   THEN GO TO S90,.                                                    BDTR1610
S80..                                                                  BDTR1620
   ID   =1,.                                                           BDTR1630
   FF   =DL1X,.                                                        BDTR1640
   DL1X =DLXX,.                                                        BDTR1650
   DLXX =FF,.                                                          BDTR1660
   XO   =1/XO,.                                                        BDTR1670
   FF   =AA,.                                                          BDTR1680
   AA   =BB,.                                                          BDTR1690
   BB   =FF,.                                                          BDTR1700
   G2   =G1,.                                                          BDTR1710
S90..                                                                  BDTR1720
   FF   =0,.                                                           BDTR1730
   IF AA LE 1                          /* TEST FOR A LESS THAN 1     */BDTR1740
   THEN DO,.                                                           BDTR1750
        CALL LGAM(AA+1,G4),.                                           BDTR1760
        DD   =AA*DLXX+BB*DL1X+G3-G2-G4,.                               BDTR1770
        IF DD GT -1.68E+2                                              BDTR1780
        THEN FF=FF+EXP(DD),.                                           BDTR1790
        AA   =AA+1,.                                                   BDTR1800
        END,.                                                          BDTR1810
   FN   =AA+BB-1,.                     /* COMPUTE P USING CONTINUED  */BDTR1820
   RR   =AA-1,.                        /* FRACTION EXPANSION         */BDTR1830
   SS   =(((BB-80)*(RR+80))/((RR+2*80-1)*(RR+2*80)))*XO,.              BDTR1840
        DO XI=79 TO 1 BY -1,.                                          BDTR1850
        DD   =((XI*(FN+XI))/((RR+2*XI+1)*(RR+2*XI)))*XO,.              BDTR1860
        CC   =(((BB-XI)*(RR+XI))/((RR+2*XI-1)*(RR+2*XI)))*XO,.         BDTR1870
        SS   =CC/(1+DD/(1-SS)),.                                       BDTR1880
        END,.                                                          BDTR1890
   SS   =1/(1-SS),.                                                    BDTR1900
   IF SS LE 0                                                          BDTR1910
   THEN GO TO S110,.                                                   BDTR1920
   CALL LGAM(AA+BB,G1),.                                               BDTR1930
   CALL LGAM(AA+1,G4),.                                                BDTR1940
   PP   =G1-G2-G4+AA*DLXX+(BB-1)*DL1X+LOG(SS),.                        BDTR1950
   IF PP LE -1.68E+2                                                   BDTR1960
   THEN DO,.                                                           BDTR1970
        PP   =FF,.                                                     BDTR1980
        GO TO S100,.                                                   BDTR1990
        END,.                                                          BDTR2000
   PP   =EXP(PP)+FF,.                                                  BDTR2010
S100..                                                                 BDTR2020
   IF ID GT 0                                                          BDTR2030
   THEN PP=1-PP,.                                                      BDTR2040
   P    =PP,.                                                          BDTR2050
   IF P LT 0                           /* SET ERROR INDICATOR        */BDTR2060
   THEN IF ABS(P) GT 1E-7                                              BDTR2070
        THEN GO TO S110,.                                              BDTR2080
        ELSE DO,.                                                      BDTR2090
             P    =0,.                                                 BDTR2100
             GO TO S130,.                                              BDTR2110
             END,.                                                     BDTR2120
   ELSE IF P GT 1                                                      BDTR2130
        THEN IF ABS(1-P) GT 1E-7                                       BDTR2140
             THEN                                                      BDTR2150
S110..                                                                 BDTR2160
                  DO,.                                                 BDTR2170
                  ERROR='3',.                                          BDTR2180
                  P    =+1E+75,.                                       BDTR2190
                  GO TO S140,.                                         BDTR2200
                  END,.                                                BDTR2210
             ELSE DO,.                                                 BDTR2220
                  P    =1,.                                            BDTR2230
                  GO TO S130,.                                         BDTR2240
                  END,.                                                BDTR2250
        ELSE                                                           BDTR2260
S120..                                                                 BDTR2270
             IF P LE 1E-8                                              BDTR2280
             THEN DO,.                                                 BDTR2290
                  P    =0,.                                            BDTR2300
                  GO TO S130,.                                         BDTR2310
                  END,.                                                BDTR2320
             ELSE IF 1-P LE 1E-8                                       BDTR2330
                  THEN P=1,.                                           BDTR2340
S130..                                                                 BDTR2350
        ERROR='0',.                                                    BDTR2360
S140..                                                                 BDTR2370
   RETURN,.                                                            BDTR2380
   END,.                               /* END OF PROCEDURE BDTR      */BDTR2390
```

Purpose:

BDTR computes P(x) = probability that the random variable X, distributed according to the beta distribution with parameters A and B, is less than or equal to x. f (A, B, X), the ordinate of the beta density of X, is also computed.

Usage:

CALL BDTR (X, A, B, P, D);

X - BINARY FLOAT
Given variable containing the scalar for which P(x) is computed.

A - BINARY FLOAT
Given variable containing the beta distribution parameter.

B - BINARY FLOAT
Given variable containing the beta distribution parameter.

P - BINARY FLOAT
Resultant variable containing the probability.

D - BINARY FLOAT
Resultant variable containing the density.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following constitute the possible error conditions that may be detected:

ERROR=1 - invalid value of X. ($X<0$ or $X>1$)

ERROR=2 - invalid value of A or B (A or B $<.5$, or A or B $> 10^5$).
If either of the above conditions exists, the values of P and D are set to -1.E75.

ERROR=3 - Invalid output ($P<0$ or $P>1$). If this condition exists, the value of P is set to 1.E75.

Subroutines and function subroutines required:

CDTR
LGAM
NDTR

Method:

Refer to:

R. E. Bargmann and S. P. Ghosh, "Statistical Description Programs for a Computer Language", IBM Research Report RC-1094, 1963.
M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions. U. S. Department of Commerce, National Bureau of Standards Applied Mathematics Series, 1966.

Mathematical Background:

This subroutine computes $P=I_x$ (m, n)=Prob ($X \leq x$), where X is a random variable following the beta distribution with degrees of freedom (continuous parameters) m and n. For computation to take place, $0 \leq x \leq 1$, $0.5 \leq m \leq 10^{+5}$, and $0.5 \leq n \leq 10^{+5}$. D, the ordinate of the beta density at x, is also presented in the output.

For $0 \leq x \leq 1$, $I_x(m, n)$ may be written as:

$$I_x(m, n) = \int_0^x f(m, n, y)\, dy$$

where:

$$f(m, n, y) = \frac{1}{B(m,n)} y^{m-1} (1-y)^{n-1} \qquad (1)$$

$$B(m,n) = \frac{\Gamma(m)\Gamma(n)}{\Gamma(m+n)};\ D = f(m,n,x)$$

$I_x(m,n)$ can be reduced to a binominal partial sum that can be evaluated by means of a continued fraction expansion.

Let N = m+n-1 and r = m-1. Then:

$$I_x(m,n) = I_x(r+1,\ N-r)$$

$$I_x(r+1,\ N-r) = \sum_{s=r+1}^{N} \binom{N}{s} x^s \ (1-x)^{N-s}$$

$$= \binom{N}{r+1} x^{r+1} (1-x)^{N-r-1} S \qquad (2)$$

where $0 \leq s \leq N$

S is a continued fraction, with 80 terms being sufficient for the desired accuracy.

$$S = \frac{1}{1-}\frac{c_1}{1+}\frac{d_1}{1-}\frac{c_2}{1+}\frac{d_2}{1-}\cdots\frac{c_{80}}{1+}\frac{d_{80}}{1} \qquad (3)$$

$$c_i = \frac{(N-i-r)\ (r+i)}{(r+2i-1)\ (r+2i)}\ \frac{x}{1-x} \qquad (4)$$

$$d_i = \frac{i\ (N+i)}{(r+2i+1)\ (r+2i)}\ \frac{x}{1-x} \qquad (5)$$

The above continued fraction expansion of $I_x(m,n)$ holds for positive m and n (integers or nonintegers), $N \geq 0$ $(m+n \geq 1)$, and $r \geq 0$ $(m \geq 1)$. In order to fulfill these last two conditions, if $m < 1$, the following transformation must be made before computation of $I_x(m,n)$ can take place:

$$I_x(m,n) = \frac{\Gamma(m+n)}{\Gamma(m+1)\Gamma(n)} x^m (1-x)^n + I_x(m+1,n) \qquad (6)$$

The quantities on the right-hand side of equation (6) are those that are computed.

It is known that $I_x(m,n) = I_{1-x}(n,m)$. Thus, either of the two parameter sets indicated by this equation may be used in computing the beta integral. The parameter set selection is made by applying the following empirically derived rule:

> Let p and q be the degrees of freedom corresponding to z, where $z = x$ if $x \leq .5$ or $(1-x)$ otherwise. If the quantity $[(p-1) - (p+q-1) z^{6/5} + 2]$ is positive, use the parameter set corresponding to z. Otherwise, use the parameter set corresponding to (1-z).

If $0 \leq x \leq 10^{-8}$ or $0 \leq 1-x \leq 10^{-8}$, the approximation is made that x = 0 or 1 respectively. P and D are then set according to the following table:

$0 \leq x \leq 10^{-8}$		$0 \leq 1-x \leq 10^{-8}$	
$P = 0$		$P = 1$	
If:	Then:	If:	Then:
$A < 1$	$D = 10^{75}$	$B < 1$	$D = 10^{75}$
$A = 1$	$D = 1/B(m,n)$	$B = 1$	$D = 1/B(m,n)$
$A > 1$	$D = 0$	$B > 1$	$D = 0$

If either m or n, or both are within 10^{-8} of 1, the beta integral is solved explicitly for m = 1, n = 1, or m = n = 1:

If:	Then:
$A = 1,\ B = 1$	$P = x$
$A = 1,\ B \neq 1$	$P = 1 - (1-x)^n$
$A \neq 1,\ B = 1$	$P = x^m$

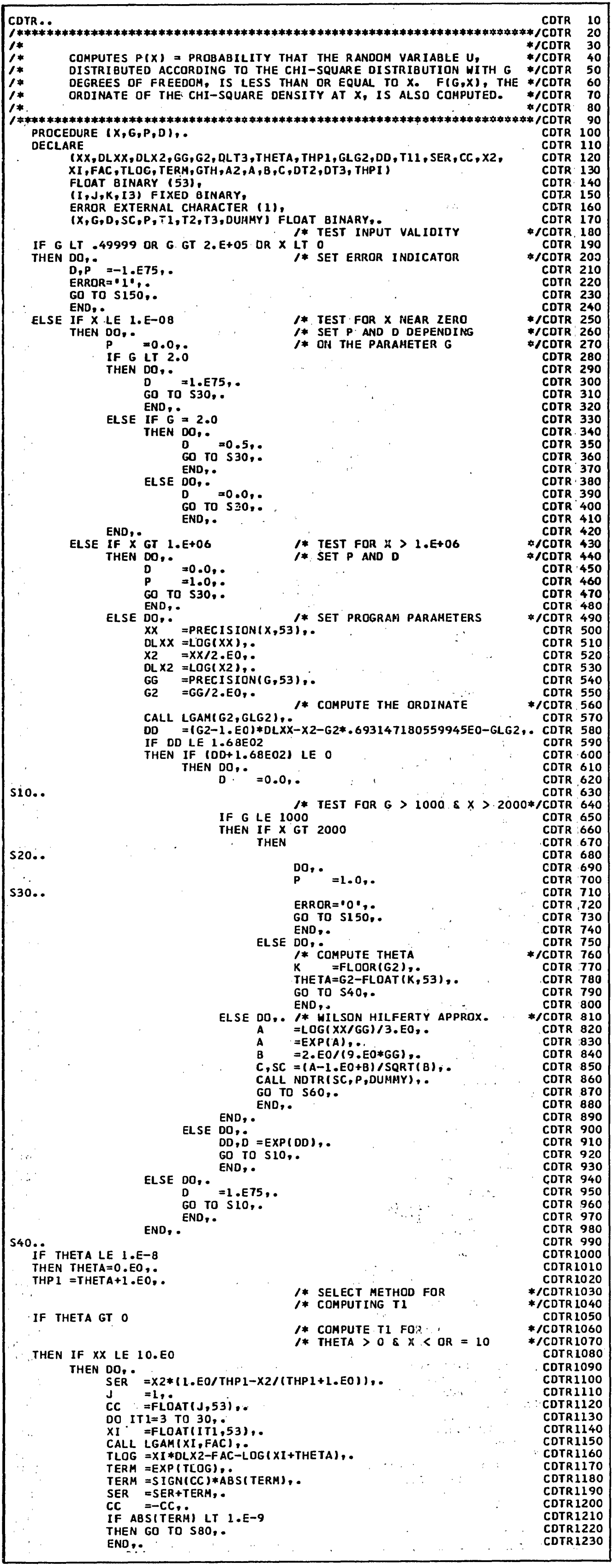
If m or n is greater than 1000, the chi-square approximation is used:

$z_1 = 2m\,(1-x)/x$ is distributed as χ^2 with 2n degrees of freedom and $P = 1 - P_{\chi^2}\,(z_1)$ for $m > 1000$.

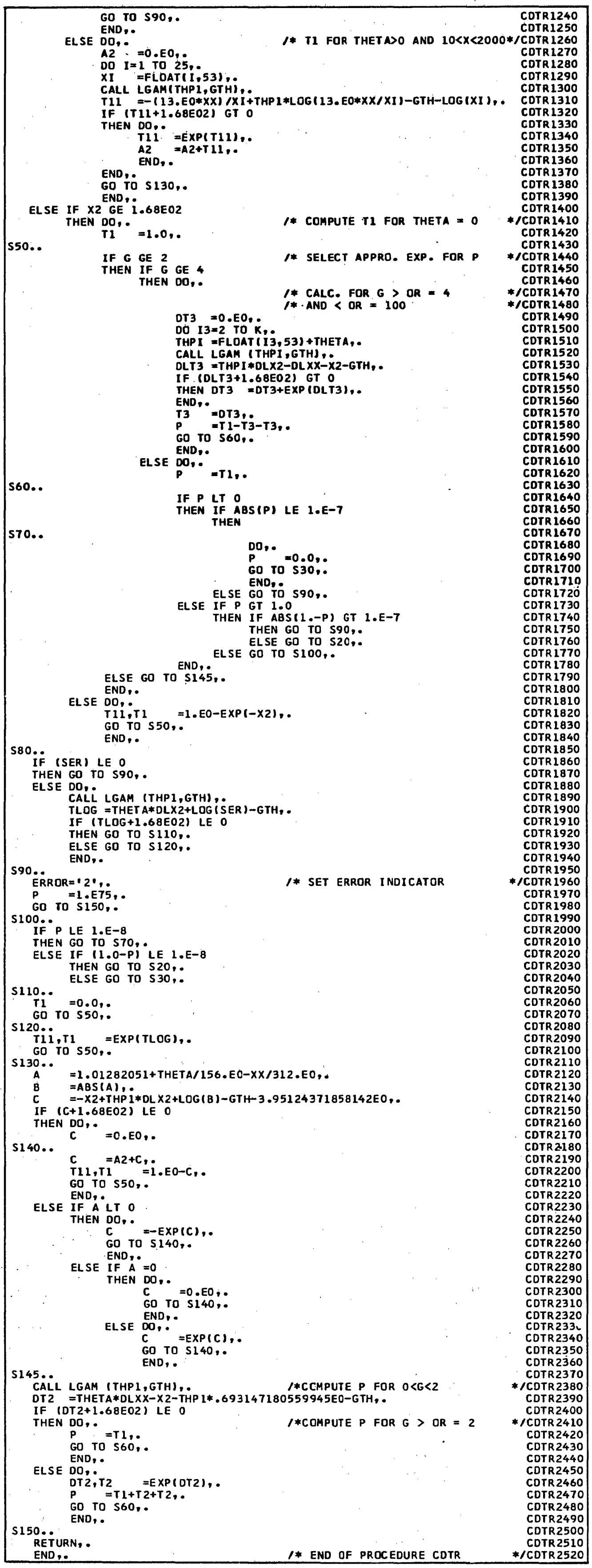
$z_2 = 2nx/\,(1-x)$ is distributed as χ^2 with 2 m degrees of freedom and $P = P_{\chi^2}\,(z_2)$ for $n > 1000$. If both m and n are greater than 1000, the approximation corresponding to z_1 is used.

The values of P very near zero or one may be somewhat imprecise. To eliminate possible misinterpretation of results, if $0 \le P \le 10^{-8}$ or $0 \le 1-P \le 10^{-8}$, P is set to 0 or 1 respectively.

- Subroutine CDTR



```
CDTR..
/*******************************************************************/
/*                                                                 */
/*      COMPUTES P(X) = PROBABILITY THAT THE RANDOM VARIABLE U,    */
/*      DISTRIBUTED ACCORDING TO THE CHI-SQUARE DISTRIBUTION WITH G */
/*      DEGREES OF FREEDOM, IS LESS THAN OR EQUAL TO X.  F(G,X), THE */
/*      ORDINATE OF THE CHI-SQUARE DENSITY AT X, IS ALSO COMPUTED.  */
/*                                                                 */
/*******************************************************************/
   PROCEDURE (X,G,P,D),.
   DECLARE
        (XX,DLXX,DLX2,GG,G2,DLT3,THETA,THP1,GLG2,DD,T11,SER,CC,X2,
        XI,FAC,TLOG,TERM,GTH,A2,A,B,C,DT2,DT3,THPI)
        FLOAT BINARY (53),
        (I,J,K,I3) FIXED BINARY,
        ERROR EXTERNAL CHARACTER (1),
        (X,G,D,SC,P,T1,T2,T3,DUMMY) FLOAT BINARY,.
                                      /* TEST INPUT VALIDITY         */
   IF G LT .49999 OR G GT 2.E+05 OR X LT 0
   THEN DO,.                          /* SET ERROR INDICATOR         */
        D,P  =-1.E75,.
        ERROR='1',.
        GO TO S150,.
        END,.
   ELSE IF X LE 1.E-08                /* TEST FOR X NEAR ZERO        */
        THEN DO,.                     /* SET P AND D DEPENDING       */
             P    =0.0,.              /* ON THE PARAMETER G          */
             IF G LT 2.0
             THEN DO,.
                  D    =1.E75,.
                  GO TO S30,.
                  END,.
             ELSE IF G = 2.0
                  THEN DO,.
                       D    =0.5,.
                       GO TO S30,.
                       END,.
                  ELSE DO,.
                       D    =0.0,.
                       GO TO S30,.
                       END,.
             END,.
        ELSE IF X GT 1.E+06           /* TEST FOR X > 1.E+06         */
             THEN DO,.                /* SET P AND D                 */
                  D    =0.0,.
                  P    =1.0,.
                  GO TO S30,.
                  END,.
             ELSE DO,.                /* SET PROGRAM PARAMETERS      */
                  XX   =PRECISION(X,53),.
                  DLXX =LOG(XX),.
                  X2   =XX/2.E0,.
                  DLX2 =LOG(X2),.
                  GG   =PRECISION(G,53),.
                  G2   =GG/2.E0,.
                                      /* COMPUTE THE ORDINATE        */
                  CALL LGAM(G2,GLG2),.
                  DD   =(G2-1.E0)*DLXX-X2-G2*.693147180559945E0-GLG2,.
                  IF DD LE 1.68E02
                  THEN IF (DD+1.68E02) LE 0
                       THEN DO,.
                            D    =0.0,.
S10..
                                      /* TEST FOR G > 1000 & X > 2000*/
                            IF G LE 1000
                            THEN IF X GT 2000
                                 THEN
S20..
                                      DO,.
                                      P    =1.0,.
S30..
                                      ERROR='0',.
                                      GO TO S150,.
                                      END,.
                                 ELSE DO,.
                                      /* COMPUTE THETA               */
                                      K    =FLOOR(G2),.
                                      THETA=G2-FLOAT(K,53),.
                                      GO TO S40,.
                                      END,.
                            ELSE DO,. /* WILSON HILFERTY APPROX.     */
                                 A    =LOG(XX/GG)/3.E0,.
                                 A    =EXP(A),.
                                 B    =2.E0/(9.E0*GG),.
                                 C,SC =(A-1.E0+B)/SQRT(B),.
                                 CALL NDTR(SC,P,DUMMY),.
                                 GO TO S60,.
                                 END,.
                            END,.
                       ELSE DO,.
                            DD,D =EXP(DD),.
                            GO TO S10,.
                            END,.
                  ELSE DO,.
                       D    =1.E75,.
                       GO TO S10,.
                       END,.
                  END,.
S40..
   IF THETA LE 1.E-8
   THEN THETA=0.E0,.
   THP1 =THETA+1.E0,.
                                      /* SELECT METHOD FOR           */
                                      /* COMPUTING T1                */
   IF THETA GT 0
                                      /* COMPUTE T1 FOR              */
                                      /* THETA > 0 & X < OR = 10     */
   THEN IF XX LE 10.E0
        THEN DO,.
             SER  =X2*(1.E0/THP1-X2/(THP1+1.E0)),.
             J    =1,.
             CC   =FLOAT(J,53),.
             DO IT1=3 TO 30,.
             XI   =FLOAT(IT1,53),.
             CALL LGAM(XI,FAC),.
             TLOG =XI*DLX2-FAC-LOG(XI+THETA),.
             TERM =EXP(TLOG),.
             TERM =SIGN(CC)*ABS(TERM),.
             SER  =SER+TERM,.
             CC   =-CC,.
             IF ABS(TERM) LT 1.E-9
             THEN GO TO S80,.
             END,.
```



```
             GO TO S90,.
             END,.
        ELSE DO,.                     /* T1 FOR THETA>0 AND 10<X<2000*/
             A2    =0.E0,.
             DO I=1 TO 25,.
             XI   =FLOAT(I,53),.
             CALL LGAM(THP1,GTH),.
             T11  =-(13.E0*XX)/XI+THP1*LOG(13.E0*XX/XI)-GTH-LOG(XI),.
             IF (T11+1.68E02) GT 0
             THEN DO,.
                  T11  =EXP(T11),.
                  A2   =A2+T11,.
                  END,.
             END,.
             GO TO S130,.
             END,.
   ELSE IF X2 GE 1.68E02
        THEN DO,.                     /* COMPUTE T1 FOR THETA = 0    */
             T1   =1.0,.
S50..
             IF G GE 2                /* SELECT APPRO. EXP. FOR P    */
             THEN IF G GE 4
                  THEN DO,.
                                      /* CALC. FOR G > OR = 4        */
                                      /* AND < OR = 100              */
                       DT3  =0.E0,.
                       DO I3=2 TO K,.
                       THPI =FLOAT(I3,53)+THETA,.
                       CALL LGAM (THPI,GTH),.
                       DLT3 =THPI*DLX2-DLXX-X2-GTH,.
                       IF (DLT3+1.68E02) GT 0
                       THEN DT3  =DT3+EXP(DLT3),.
                       END,.
                       T3   =DT3,.
                       P    =T1-T3-T3,.
                       GO TO S60,.
                       END,.
                  ELSE DO,.
                       P    =T1,.
S60..
                       IF P LT 0
                       THEN IF ABS(P) LE 1.E-7
                            THEN
S70..
                                 DO,.
                                 P    =0.0,.
                                 GO TO S30,.
                                 END,.
                            ELSE GO TO S90,.
                       ELSE IF P GT 1.0
                            THEN IF ABS(1.-P) GT 1.E-7
                                 THEN GO TO S90,.
                                 ELSE GO TO S20,.
                            ELSE GO TO S100,.
                       END,.
             ELSE GO TO S145,.
             END,.
        ELSE DO,.
             T11,T1    =1.E0-EXP(-X2),.
             GO TO S50,.
             END,.
S80..
   IF (SER) LE 0
   THEN GO TO S90,.
   ELSE DO,.
        CALL LGAM (THP1,GTH),.
        TLOG =THETA*DLX2+LOG(SER)-GTH,.
        IF (TLOG+1.68E02) LE 0
        THEN GO TO S110,.
        ELSE GO TO S120,.
        END,.
S90..
   ERROR='2',.                        /* SET ERROR INDICATOR         */
   P    =1.E75,.
   GO TO S150,.
S100..
   IF P LE 1.E-8
   THEN GO TO S70,.
   ELSE IF (1.0-P) LE 1.E-8
        THEN GO TO S20,.
        ELSE GO TO S30,.
S110..
   T1   =0.0,.
   GO TO S50,.
S120..
   T11,T1    =EXP(TLOG),.
   GO TO S50,.
S130..
   A    =1.01282051+THETA/156.E0-XX/312.E0,.
   B    =ABS(A),.
   C    =-X2+THP1*DLX2+LOG(B)-GTH-3.95124371858142E0,.
   IF (C+1.68E02) LE 0
   THEN DO,.
        C    =0.E0,.
S140..
        C    =A2+C,.
        T11,T1    =1.E0-C,.
        GO TO S50,.
        END,.
   ELSE IF A LT 0
        THEN DO,.
             C    =-EXP(C),.
             GO TO S140,.
             END,.
        ELSE IF A =0
             THEN DO,.
                  C    =0.E0,.
                  GO TO S140,.
                  END,.
             ELSE DO,.
                  C    =EXP(C),.
                  GO TO S140,.
                  END,.
S145..
   CALL LGAM (THP1,GTH),.             /*CCMPUTE P FOR 0<G<2          */
   DT2  =THETA*DLXX-X2-THP1*.693147180559945E0-GTH,.
   IF (DT2+1.68E02) LE 0
   THEN DO,.                          /*COMPUTE P FOR G > OR = 2     */
        P    =T1,.
        GO TO S60,.
        END,.
   ELSE DO,.
        DT2,T2    =EXP(DT2),.
        P    =T1+T2+T2,.
        GO TO S60,.
        END,.
S150..
   RETURN,.
   END,.                              /* END OF PROCEDURE CDTR       */
```

Purpose:

CDTR computes P(x) = the probability that the random variable X, distributed according to the chi-square distribution with G degrees of freedom, is less than or equal to x. f (G x), the ordinate of the chi-square density at x, is also computed.

Usage:

CALL CDTR (X, G, P, D);

X - BINARY FLOAT
Given random variable following the chi-square distribution.

G - BINARY FLOAT
Given variable containing the number of degrees of freedom of the chi-square distribution. G is a continuous parameter such that $.5 \leq G \leq 2\ (10^5)$.

P - BINARY FLOAT
Resultant variable containing the probability.

D - BINARY FLOAT
Resultant variable containing the density.

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. The following consitute the possible error conditions that may be detected:

ERROR=1 - invalid value of X.
$(X < 0)$ or invalid value of G.
$(G < .5$ or $G > 200,000)$
If this condition exists, the values of P and D are set to -1.E75.

ERROR=2 - invalid output $(P < 0$ or $P > 1)$ or the series T1 has failed to converge. If this condition exists, the values of P and D are set to -1.E75.

Subroutines and function subroutines required:

LGAM
NDTR

Method:

For reference see:

1. R. E. Bargmann and S. P. Ghosh, "Statistical Distribution Programs for a Computer Language", IBM Research Report RC-1094, 1963.
2. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions U. S.

Department of Commerce, National Bureau of Standards Applied Mathematics Series, 1966.

Mathematical Background:

This subroutine computes P=P(x)=Prob. ($X \leq x$), where X is a random variable following the χ^2 distribution with continuous parameter g. X must be greater than or equal to zero and $.5 \leq g \leq 2\ (10^5)$ for computation to take place. D, the ordinate of the χ^2 density at x, is also presented in the output.

For $x \geq 0$, P(x) may be written as:

$$P(x) = \int_0^x f(g, y)\, dy \qquad (1)$$

where:

$$f(g, y) = y^{(g-2)/2} e^{-y/2} \Big/ \left(2^{g/2}\, \Gamma\left(\frac{g}{2}\right)\right)$$

$$D = f(g, x)$$

To evaluate the integral, we first define

$$\theta = \frac{g}{2} - \left[\frac{g}{2}\right],$$

where $\frac{g}{2}$ denotes the largest integer less than or equal to $\frac{g}{2}$. θ is thus the fractional part of $\frac{g}{2}$.

Substituting this expression into the integral and performing the proper reductions, we find:

If:	Then:
$0 < g < 2$	$P(x) = T1 + T2$
$2 \leq g < 4$	$P(x) = T1$
$g \geq 4$	$P(x) = T1 - 2T3$

where:

$$T1 = \int_0^x \frac{y^\theta e^{-y/2}\, dy}{2^{1+\theta}\, \Gamma(1+\theta)}$$

$$T2 = f(2+2\theta, x)$$

$$T3 = \sum_{i=2}^{\left[\frac{g}{2}\right]} f(2i+2\theta, x)$$

T2 and T3 may be evaluated directly using logs and antilogs.

If $\theta = 0$ ($\frac{g}{2}$ is an integer), T1 is easily evaluated as:

$$T1 = 1 - e^{-x/2}$$

If $\theta > 0$, T1 can be expanded in the following infinite series:

$$T1 = \frac{Z^\theta}{\Gamma(1+\theta)} \left\{ \frac{Z}{1+\theta} - \frac{Z^2}{2+\theta} + \frac{Z^3}{2!(3+\theta)} - \frac{Z^4}{3!(4+\theta)} \cdots \right\} \qquad (2)$$

where $Z = \frac{x}{2}$.

This series is used in the range $10^{-8} < x \leq 10$, and not more than 30 terms are necessary to ensure convergence within error bounds of 10^{-9}.

For $x > 10$, 1-T1 is evaluated by the Euler-McLaurin formula up to third derivative terms (see Reference 2, equation 23.1.30). One finds:

$$1 - T1 = \int_0^N h(u)\, du \qquad (3)$$

where:

$$h(u) = \frac{1}{\Gamma(1+\theta)} \left(\frac{2u}{Nx}\right)^{-(1+\theta)} u^{-1} e^{-Nx/2u}$$

$$\int_0^N h(u)\, du = \sum_{u=0}^{N-1} h(u) + \frac{1}{2} h(N) - \frac{1}{12} h'(N) + \frac{1}{720} h'''(N)$$

(Note: $h' = h''' = 0$ at 0.)

In order to achieve accuracy consistent with that obtained by the method of equation (2), N=26 is used in equation (3).

If $0 \leq x \leq 10^{-8}$, the approximation is made that x=0. P is set to 0, and D is set to 1.E75, .5, or 0, corresponding to g less than 2, equal to 2, or greater than 2 respectively.

If $g > 1000$, Wilson and Hilferty's approximation is used. $\left(\frac{x^2}{g}\right)^{1/3}$ is approximately normally distributed with mean $1 - \frac{2}{9g}$ and variance $\frac{2}{9g}$ (see reference 2, equation 26.4.14). If $g \leq 1000$ and $x > 2000$, or $g > 1000$ and $x > 10^6$, P is set to 1.

Since T1 may have an error of about 10^{-9}, values of P(x) very near zero or one may be somewhat imprecise. To eliminate possible misinterpretation of results, if $0 \le P(x) \le 10^{-8}$ or $0 \le 1 - P(x) \le 10^{-8}$, P(x) is set to 0 or 1 respectively.

The X^2 distribution is a member of the gamma family of probability distributions. The general form for distributions of this class is:

$$P_G(x) = \int_0^x G\ (n, A,\ \Psi; u)\ du$$

where

$$G\ (n,\ a,\ \Psi;\ u) = (u-a)^{n-1}\ e^{-(u-a)/\Psi} \Big/ \left(\Psi^n \Gamma\ (n) \right).$$

This subroutine may, therefore, also be used to compute the probability integral from zero to x and the corresponding ordinate at x for any member of this gamma family by setting:

$$x = 2(u-a)\ /\ \Psi \text{ and } g = 2n$$

Then P(x) will be the desired probability, and $\frac{2f(g, x)}{\Psi}$ will be the desired ordinate.

● Subroutine NDTI

```
NDTI..
/********************************************************************/
/*                                                                  */
/*      COMPUTES X=P**X(-1)(Y), THE ARGUMENT X SUCH THAT Y=P(X)=THE */
/*      PROBABILITY THAT THE RANDOM VARIABLE U, DISTRIBUTED NORMALLY */
/*      (0,1), IS LESS THAN OR EQUAL TO X.  F(X) THE ORDINATE OF THE */
/*      NORMAL DENSITY, AT X, IS ALSO COMPUTED.                     */
/*                                                                  */
/********************************************************************/
   PROCEDURE(P,X,D),.
   DECLARE
        (P,X,D,T2,T) FLOAT BINARY,
        ERROR EXTERNAL CHARACTER (1),.
   ERROR='0',.
   X,D  =0,.
   IF P LT 0.0
   THEN
S10..
        DO,.                        /* P < 0--SET ERROR INDICATOR  */
        ERROR='1',.
        GO TO S30,.
        END,.
   ELSE IF P = 0.0
        THEN DO,.                   /* P = 0--SET X AND D          */
             X    =-.999999E+74,.
S20..
             D    =0.0,.
             GO TO S30,.
             END,.
        ELSE IF P GT 1.0
             THEN GO TO S10,.       /* P > 1--SET ERROR INDICATOR  */
             ELSE IF P = 1.0
                  THEN DO,.         /* P = 1--SET X AND D          */
                       X    =.999999E+74,.
                       GO TO S20,.
                       END,.
                  ELSE DO,.         /* P > 0 AND P < 1             */
                       D    =P,.
                       IF D GT 0.5
                                    /* COMPLEMENT P                */
                       THEN D    =1.0-D,.
                                    /* CALC. EQUATION 2 IN WRITE UP*/
                       T2   =LOG(1.0/(D*D)),.
                       T    =SQRT(T2),.
                                    /* CALC. EQUATION 1 IN WRITE UP*/
                       X    =T-(2.515517+0.802853*T+0.010328*T2)/
                             (1.0+1.432788*T+0.189269*T2+0.001308*T
                             *T2),.
                       IF P LE 0.5  /* P < OR = .5                 */
                       THEN X    =-X,./* NEGATE X                  */
                                    /* CALCULATE DENSITY           */
                       D    =0.3989423*EXP(-X*X/2.0),.
                       END,.
S30..
   RETURN,.
   END,.                            /* END OF PROCEDURE NDTI       */
```

Purpose:

NDTI computes $x = P^{-1}$ (y) such that y = P(x), the probability that the random variable X, distributed normally (0, 1) is less than or equal to x. f(x), the ordinate of the normal density at x, is also computed.

Usage:

CALL NDTI (P, X, D);

P - BINARY FLOAT
Given variable containing the probability.

X - BINARY FLOAT
Resultant variable such that P=Y= the probability that u, the random variable, is less than or equal to X.

D - BINARY FLOAT
Resultant variable containing the density f (X).

Remarks:

If no errors are detected in the processing of data, the error indicator, ERROR, is set to zero. However, if P=0, X is set to $-(10)^{74}$, and D is set to zero. If P=1, X is set to $(10)^{74}$ and D is set to zero. The following constitutes the possible error condition that may be detected:

ERROR=1 - Invalid value of P. P is either less than zero or greater than one.

Method:

Refer to:

C. Hastings, Approximations for Digital Computers, Princeton University Press, Princeton, N. J., 1955.

M. Abramowitz and Stegun, I. A. Handbook of Mathematical Functions, Dover Publications, Inc., N.Y., equation 26.2.23.

Mathematical Background:

This subroutine computes $x = P^{-1}(y)$ such that $y = P(x) = \text{Prob}(X \leq x)$, where X is a random variable distributed normally with mean zero and variance one. That is, given P(x), the following is solved for x:

$$P(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-u^2/2)\, du$$

The following approximation is used:

$$x = w - \sum_{i=0}^{2} a_i w^i / \sum_{i=0}^{3} b_i w^i \qquad (1)$$

where:

$$w = \sqrt{\ln (1/p^2)} \quad (0 < p \leq .5) \qquad (2)$$

$a_0 = 2.515517$

$a_1 = 0.802853$

$a_2 = 0.010328$

$b_1 = 1.432788$

$b_2 = 0.189269$

$b_3 = 0.001308$

If P(x) is greater than 0.5, 1-P(x) is used as p in (2) above, and the result of (1), x, is negated. The maximum error is 0.00045; f(x) is also calculated.

APPENDIX A: ACCURACY OF SUBROUTINES

The subroutines in SSP can be broken down into three major categories from the standpoint of accuracy:

(1) those having little or no effect on accuracy,
(2) those whose accuracy depends on the characteristics of the input data, and
(3) those in which definite statements on accuracy can be made.

SUBROUTINES WITH LITTLE OR NO EFFECT 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.

ABST
BUND
CHSQ
HTES
KLM2
KRNK
MOMN
MPIT
MPRM
MTPI
MSCG
MSCS
ORDR
QTST
RANK
SRNK
SUBM
SBST
TAB1
TAB2
TALY
TIE
TRAC
TTST
TWAV
UTST
WTST

SUBROUTINES WITH DATA-DEPENDENT ACCURACY

The accuracy of the following subroutines cannot be predicted because it depends 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 analysis considerations. It cannot be assumed that the results are accurate simply because subroutine execution is completed.

ACFM/ACFE
AHIM/AHIE
ALIM/ALIE
APCI/APC2
APLL
ASN
AVAR
CANC
CORR
DERE
DET3
DET5
DFEC
DFEO
DGT3
DMTX
DSCR
EXSM
FFT
FFTM
FMFP
KLMO
LOAD
MAGS
MATE
MATU
MDLG
MDLS/MDRS
MDSB
MEAT
MEBS
MEST
MFG
MFGR
MFS
MFSB
MGB1/MGB2
MGDU
MIG
MINV
MIS
MLSQ
MLTR
MMGG
MMGS
MMGT
MMSS
MSDU
MSTU
MVAT
MVEB
MVST
MVSU
MVUB
PEC/PTC
POST
POSV
PRTC
PRTR
QA2
QA4
QA8
QA12
QA16
QA24
QATR
QG2
QG4
QG8
QG16
QG24
QG32
GH2
QH4
QH8
QH16
QH24
QH32
QH48
QHFG/QHFE/
QHSQ/QHSE
QL2
QL4
QL8
QL12
QL16
QL24
QSF
QTFG/QTFE
RTF
RTFD
SE15
SE35
SG13 /SE13
STRG
VRMX

SUBROUTINES WITH DEFINITE ACCURACY CHARACTERISTICS

The subroutines in this section have accuracy characteristics that can be specified on an individual basis. The mathematical descriptions for many of these subroutines contain information on truncation error of a strictly theoretical nature. The actual implementation of these subroutines on System/360 results in the accuracy noted in the following table. The standard reference for comparing the accuracy of these subroutines is M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards, Washington, D.C., March 1965. However, in certain cases, other tables were used, as noted below. It should be remembered that in System/360 single-precision floating point, there are just over six significant figures.

Maximum differences below are given in terms of number of decimal places (DP) and/or number of significant digits (SD) that agree. The number of digits tabled should be considered when accuracy statements are viewed; that is, certain tables are given to only five places, whereas the algorithms used may be more accurate. In compiling maximum differences, the maximum was taken over the set of points indicated in the table. The average difference was normally much smaller.

The notation x = a (b) c implies that a, a + b, a + 2b, . . . , c were the arguments (x) used.

Name	Functions calculated	Functions checked with reference	Range checked with reference	Maximum difference

BDTR	$p=I_x\,(a,b)$	$I_x^{-1}\,(a,b)$ Tables by Leon H. Harter: New Tables of the Incomplete Gamma Function Ratio and of Percentage Points of the χ^2 and Beta Distribution, 1964	p=.0001, .0005
			a=1 (1) 40
			b=5(5) 40
			p=.0100, .0500
			a= 2 (2) 10
			b=5 (5) 30
				correct to 5 DP

CDTR	$y=P_g\,(x)$ where P is the χ^2 distribution function with parameter g.	$y = P_g\,(x)$	x=.001 (.001) .01;
			.01 (.01) 1.0;
			1.0 (.1) 2.0;
			2.0 (.2) 10.0;
			10.0 (.5) 20.0;
			20 (1) 40
			40 (2) 76
			for
			g=1 (1) 30
				1 in the 5th DP

CELI Complete elliptic 1st integral	K(k) (single precision)	K(m); $k=\sqrt{m}$	m=.01 (.01).99	2 in 7th SD
		K(α); k= sin α	α= 1(1)73	2 in 7th SD
		(α in degrees)	α= 74(1)86	3 in 7th SD
	K(k) (double precision)	K(m); $k=\sqrt{m}$	m = .01(.01).86	1 in 16th SD
			m = .87(.01) .96	4 in 16th SD
			m = .97(.01).99	11 in 16th SD
		K(α); k = sin α	α = 1(1)75	1 in 16th SD
		(α in degrees)	α = 76(1)80	2 in 16th SD
			α = 81(1)86	11 in 16th SD

Name	Functions calculated	Functions checked with reference	Range checked with reference	Maximum difference
CEL2 Generalized complete elliptic 2nd integral	K(k) with A = B = 1 E(k) with A = 1 $B = 1 - k^2$ (single precision)	K(m); $k = \sqrt{m}$	m = .01(.01) .99	2 in 7th SD
		K(α); k = sin α (α in degrees)	α = 1(1) 73	2 in 7th SD
			α = 74(1)86	3 in 7th SD
		E(m); $k = \sqrt{m}$	m = .01(01)	2 in 7th SD
		E(α); k = sin α	α = 1(1) 86	2 in 7th SD
		K'E + E'K - KK'	m = .01(.01).99	7 in 7th SD
		(Legendre's relation)	α = 1(1) 89	1 in 6th SD
	K(k) with A = B = 1 E(k) with A = 1 $B = 1 - k^2$ (double precision)	K(m); $k = \sqrt{m}$	m = .01(.01).99	2 in 16th SD
		K(α); k = sin α (α in degrees)	α = 1(1)80	2 in 16th SD
			α = 81(1)86	11 in 16th SD
		E(α); k = sin α	α = 1(1)89	2 in 16th SD
		K'E + E'K - KK' (Legendre's relation)	m = .01(.01).99	9 in 16th SD
			α = 1(1)89	9 in 16th SD

ELI1
Incomplete elliptic 1st integral
	F(ζ/α)
	with
	x = tan ζ
	k = sin α
	$ck = \sqrt{1-k^2}$
	(single precision)
		F(ζ/α)
		(ζ, α in degrees)
			ζ = 0(5)10
			α = 0(2)90
				2 in 7th DP

			ζ = 15(5)35 α = 0(2)90	7 in 7th SD
			ζ = 40(5)50 α = 0(2)90	11 in 7th DP
			ζ = 55(5)85 α = 0(2)90	3 in 7th SD

	F(φ/α)
	with
	x = tan φ
	k = sin α
	$ck = \sqrt{1 - k^2}$
	(double precision)
		F(φ/α)
		(φ, α in degrees)
			φ = 0(5)85
			α = 0(2)90
				1 in 9th DP (probably due to rounding errors in table)

		$F(\varphi/\alpha) + F(\psi/\alpha)$. $= F\left(\frac{\pi}{2}/\alpha\right)$ (φ, α, ψ in degrees)	φ = 0(5)85 α = 0(2)80 ψ = arctan f f = 1/(cos $\alpha \cdot$tan φ)	2 in 15th DP

Name	Functions calculated	Functions checked with reference	Range checked with reference	Maximum difference
ELI2 Generalized incomplete elliptic 2nd integral	$F(\zeta/\alpha)$ with A = B = 1 $E(\zeta/\alpha)$ with A = 1 and $B = 1 - k^2$ $x = \tan \zeta$ $k = \sin \alpha$ $ck = \sqrt{1 - k^2}$ (single precision)	$F(\zeta/\alpha)$ (ζ, α in degrees)	$\zeta = 0(5)10$ $\alpha = 0(2)90$	2 in 7th DP
			$\zeta = 15(5)35$ $\alpha = 0(2)90$	7 in 7th SD
			$\zeta = 40(5)50$ $\alpha = 0(2)90$	11 in 7th DP
			$\zeta = 55(5)85$ $\alpha = 0(2)90$	3 in 7th SD
		$E(\zeta/\alpha)$ (ζ, α in degrees)	$\zeta = 0, 5$ $\alpha = 0(2)90$	2 in 7th DP
			$\zeta = 10(5)35$ $\alpha = 0(2)90$	7 in 7th SD
			$\zeta = 40(5)55$ $\alpha = 0(2)90$	12 in 7th DP
			$\zeta = 60(5)85$ $\alpha = 0(2)90$	36 in 7th DP
	$F(\varphi/\alpha)$ with A = B = 1 $E(\varphi/\alpha)$ with A = 1 $B = 1-k^2$ and $x = \tan \varphi$ $k = \sin \alpha$ $ck = \sqrt{1-k^2}$ (double precision)	$F(\varphi/\alpha)$ (φ, α in degrees)	$\varphi = 0(5)85$ $\alpha = 0(2)90$	1 in 9th DP (probably due to rounding errors in table)
		$E(\varphi/\alpha)$ (φ, α in degrees)	$\varphi = 0(5)85$ $\alpha = 0(2)90$	1 in 9th DP (probably due to rounding errors in table)
		$E(\varphi/\alpha) + E(\psi/\alpha)$ $= E\left(\frac{\pi}{2}/\alpha\right) + \sin^2\alpha \sin \phi \sin \psi$ (φ, α in degrees)	$\varphi = 0(5)85$ $\alpha = 0(2)90$ $\psi = \arctan f$ $f = 1/(\cos \alpha \cdot \tan \varphi)$	2 in 15th DP
		$F(\varphi/\alpha) + F(\psi/\alpha)$ $= F\left(\frac{\pi}{2}/\alpha\right)$ (φ, α in degrees)	$\varphi = 0(5)85$ $\alpha = 0(2)82$ $\psi = \arctan f$ $f = 1/(\cos \alpha \cdot \tan \varphi)$	3 in 15th DP

Name	Functions calculated	Functions checked with reference	Range checked with reference	Maximum difference
JELF Jacobian elliptic functions	sn u = sin φ cn u = cos φ dn u $= \sqrt{1-k^2 \sin^2\varphi}$ with φ = am u or u = F(φ/α), k = sin α sck = 1 - k^2 (single precisiòn)	sn u = sin φ (φ, α in degrees)	φ = 0(1)89 α = 0(5)85	1 in 6th DP +
		cn u = cos φ (φ, α in degrees)	φ = 0(1)89 α = 0(5)85	2 in 6th DP +
		dn u $=\sqrt{1-k^2\sin^2\varphi}$ (φ, α in degrees)	φ = 0(1)89 α = 0(5)85	1 in 6th DP +

		sn u	k^2 = .00(.05).95
			t = 0(1)25
			u = t.K(k)/25
				1 in 6th DP ++

		cn u	k^2 = .00(.05).95
			t = 0(1)25
			u = t.K(k)/25
				2 in 6th DP ++

		dn u	k^2 = .00(.05).95
			t = 0(1)25
			u = t.K(k)/25
				1 in 6th DP ++

		sn u - sn(2K-u)	k^2 = .00(.05).90
			t = 0(1)25
			u = t.K(k)/25
				6 in 6th DP

		sn u + sn(2K + u)		6 in 6th DP
		sn u + sn(4K - u)		10 in 6th DP

		cn u + cn(2K - u)	k^2 = .00(.05).90
			t = 0(1)25
			u = t.K(k)/25
				4 in 6th DP

		cn u + cn(2K + u)		4 in 6th DP
		cn u - cn(4K - u)		6 in 6th DP

		dn u - dn(2K - u)	k^2 = .00(.05).90
			t = 0(1)25
			u = t.K(k)/25
				3 in 6th DP

		dn u - dn(2K + u)		3 in 6th DP
		dn u - dn(4K - u)		5 in 6th DP

+ Calculation of u = F(φ/α) with double-precision subroutine
++ Difference between result of single- and double-precision routines

Name	Functions calculated	Functions checked with reference	Range checked with reference	Maximum difference
	sn u = sin φ	sn u = sin φ (φ, α in degrees)	φ = 5(5)85 α = 0(2)90	2 in 15th DP +
Jacobian elliptic functions	cn u = cos φ dn u = $\sqrt{1 - k^2 \alpha}$	cn u = cos φ (φ, α in degrees)	φ = 5(5)85 α = 0(2)90	3 in 15th DP +
	($\alpha = \sin^2 \varphi$) with	dn u = $\sqrt{1 - k^2 \sin^2 \varphi}$ (φ, α in degrees)	φ = 5(5)85 α = 0(2)90	2 in 15th DP +

	φ = am u
	u = F(φ/α)
	k = sin α
	sck = 1 - k^2
		sn u - sn(2K - u)	k^2 = .00(.05).90
			t = 0(1)25
				5 in 15th DP

	(double precision)	sn u + sn(2K + u)	u = t.K(k)/25	5 in 15th DP
		sn u + sn(4K - u)		12 in 15th DP
		cn u + cn(2K - u)	k^2 = .00(.05).90 t = 0(1)25	3 in 15th DP
		cn u + cn(2K + u)	u = t.K(k)/25	3 in 15th DP
		cn u - cn(4K - u)		7 in 15th DP
		dn u - dn(2K - u)	k^2 = .00(.05).90 t = 0(1)25	3 in 15th DP
		dn u - dn(2K + u)	u = t.K(k)/25	2 in 15th DP
		dn u - dn(4K - u)		6 in 15th DP
LGAM (log of the gamma function)	ln Γ (x)	ln Γ (x)	x=1	6 in 9th DP
			x=1.005(.005) 1.025	9 in 8th DP
			x=1.980(.005) 1.995	9 in 8th SD
			x=1.03(.01)1.31	8 in 9th SD
			x=1.32(.01)1.67	8 in 10th SD
			x=1.68(.01)1.97	7 in 9th SD
			x=2	6 in 9th SD
		$\log_{10} \Gamma$ (x)	x=3.0(1.0)100.0	No error in 8 place tables

NDTR	y = P (x)
	P = normal
	pdf
		y = P (x)	x = -6 (.01)6	7 in 7th DP

NDTI	x = P^{-1} (y)
	p = normal
	pdf
		x = P^{-1}(y)	y =.01(.01).99	5 in 4th DP

Name	Functions calculated	Functions checked with reference	Range checked with reference	Maximum difference
SMIR Kolmogorov-Smirnov limiting distribution	L(x)	L (x); Tables by N. Smirnov, reprinted in Annals of Math. Stat. 19, pp. 280-281 (6- and 7- place tables). Double-precision version differences are given in parentheses in the right-hand column.	x = 0(.01) .61	1 in 6th DP (1 in 6th DP)
			x = .62	3 in 5th DP (see program comments) (3 in 5th DP)
			x = .63 (.01) 1.04	3 in 6th DP (2 in 6th DP)
			x = 1.05(.01)1.15	6 in 6th DP (2 in 6th DP)
			x =1.16(.01) 1.20	9 in 6th DP (2 in 6th DP)
			x = 1.21 (.01) 1.45	8 in 6th DP (3 in 6th DP)
			x=1.46(.01) 1.65	6 in 6th DP (1 in 6th DP)
			x =1.66(.01) 1.86	2 in 6th DP (0 in 6th DP)
			x =1.87	2 in 5th DP (2 in 5th DP)
			x =1.88 (.01) 2.04	2 in 6th DP (1 in 6th DP)
			x =2.05 (.01) 2.50	1 in 6th DP (1 in 6th DP)
			x =2.51 (.01) 3.5	2 in 7th DP (1 in 7th DP)

APPENDIX B: SAMPLE PROGRAM DESCRIPTIONS

The following programs are intended to exemplify linkage of subroutines within SSP/PL/I. These programs are only examples and are not meant to be representative of the state of the art.

When supplying data for the sample programs, the user is reminded that all fixed point numbers must be right-adjusted and that all floating point numbers may appear anywhere in the field, provided the decimal point is included.

The necessary job control and process cards are included in the sample programs but are not separately shown in the deck setup illustrations.

Note that arrays are limited, for each dimension, to an upper bound of 32,767.

DATA SCREENING DACR

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, DACR, a special input routine DAT1, and three subroutines from the Scientific Subroutine Package: SBST, TAB1, and BOOL. There is also one special plotting routine, HIST. For a description of subroutine BOOL see subroutine SBST.

Capacity

1. Up to 4999 observations
2. Up to 70 variables
3. Up to 99 conditions (with the existing subroutine BOOL only two conditions are considered).
4. Up to 10 data cards per observation

Input

Control Cards

A parameter card with the following format must precede each matrix of observations.

Columns	Contents	For Sample Problem
1-6	Problem number (may be alphameric)	SAMPLE
7-11	Number of observations	0100
12-16	Number of variables	0004
17-21	Number of conditions	02
22-26	Number of selections	00003
27-31	Number of data cards per observation	01

Data Cards

1. For the observation matrix, data cards have seven fields of ten columns each. The decimal point may appear anywhere in a field. If no decimal point is included, it is assumed that the decimal point is to the right of the last digit. 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 a new card.

2. For the condition matrix three ten-column fields are used. The first contains the variable number (right-justified); the next, the relations code; and the last, a floating point number that relates to the condition.

Selection Card

For each selection there will be a new selection card. The card is prepared as follows:

Columns	Contents	For Sample Problem
1-5	Number of the variable to be tabulated	00003
6-15	Lower bound	120.
16-25	Number of intervals*	20.
26-35	Upper bound	210.

The number of selection cards must agree with the value of the selection indicator, which appears in columns 22-26 of the control card.

*In the number of intervals, it should be noted that two extra intervals must be specified for those elements that fall below the lower bound and those that fall above the upper bound.

Deck Setup

The deck setup is shown in Figure 11.

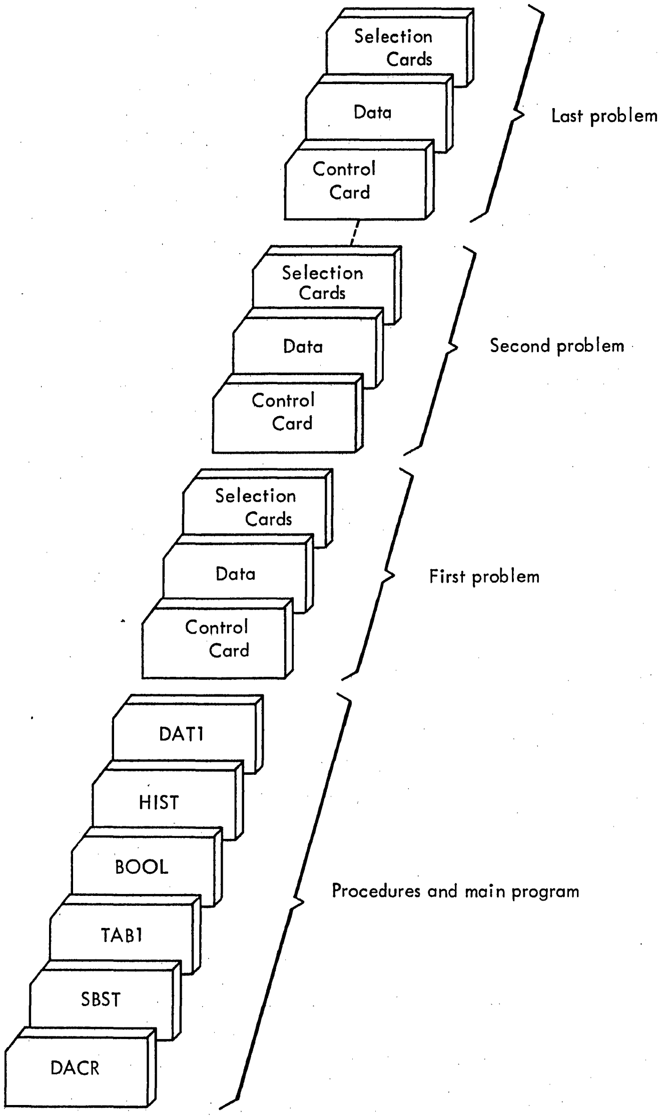
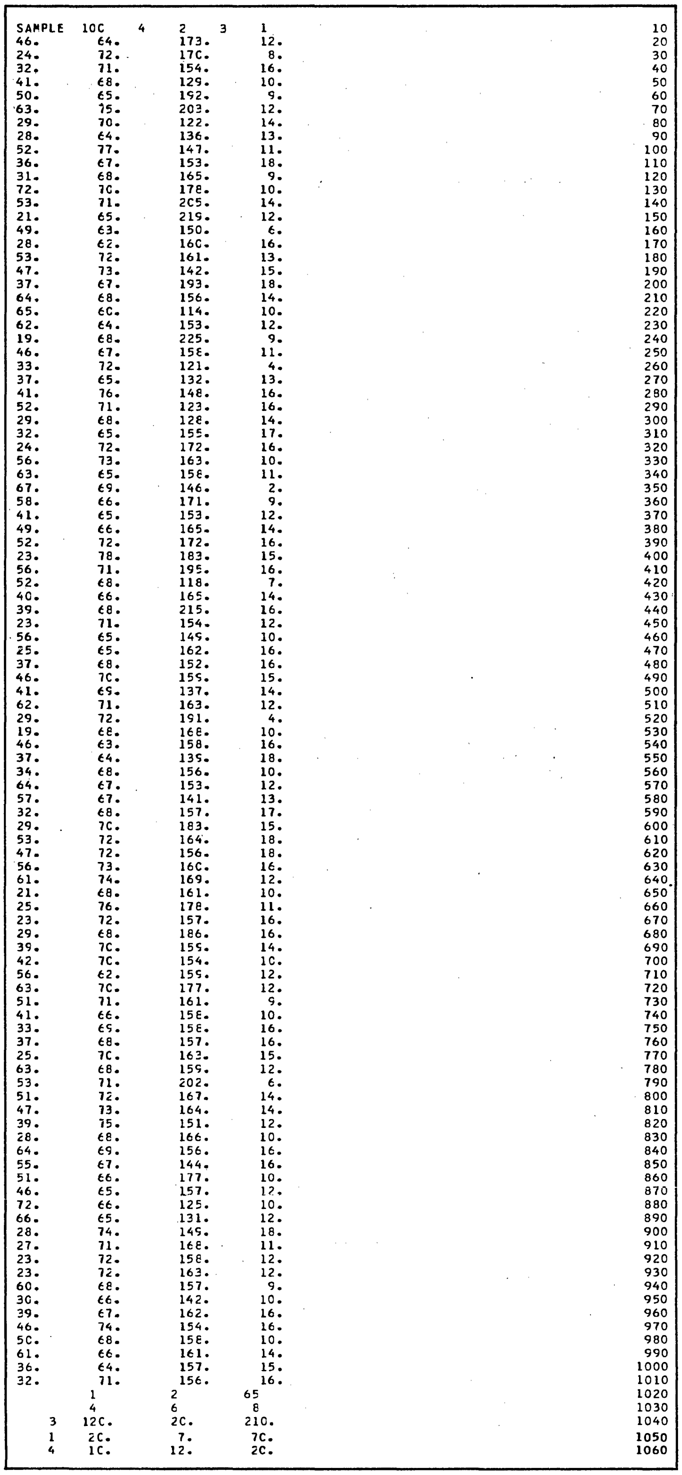


Figure 11.

Sample

A listing of input cards for the sample problem is shown in Figure 12.



```
SAMPLE  10C   4     2     3    1         10
46.        64.       173.       12.      20
24.        72.       17C.        8.      30
32.        71.       154.       16.      40
41.        68.       129.       10.      50
50.        65.       192.        9.      60
63.        75.       203.       12.      70
29.        70.       122.       14.      80
28.        64.       136.       13.      90
52.        77.       147.       11.     100
36.        67.       153.       18.     110
31.        68.       165.        9.     120
72.        7C.       178.       10.     130
53.        71.       2C5.       14.     140
21.        65.       219.       12.     150
49.        63.       150.        6.     160
28.        62.       16C.       16.     170
53.        72.       161.       13.     180
47.        73.       142.       15.     190
37.        67.       193.       18.     200
64.        68.       156.       14.     210
65.        6C.       114.       10.     220
62.        64.       153.       12.     230
19.        68.       225.        9.     240
46.        67.       158.       11.     250
33.        72.       121.        4.     260
37.        65.       132.       13.     270
41.        76.       148.       16.     280
52.        71.       123.       16.     290
29.        68.       128.       14.     300
32.        65.       155.       17.     310
24.        72.       172.       16.     320
56.        73.       163.       10.     330
63.        65.       158.       11.     340
67.        69.       146.        2.     350
58.        66.       171.        9.     360
41.        65.       153.       12.     370
49.        66.       165.       14.     380
52.        72.       172.       16.     390
23.        78.       183.       15.     400
56.        71.       195.       16.     410
52.        68.       118.        7.     420
40.        66.       165.       14.     430
39.        68.       215.       16.     440
23.        71.       154.       12.     450
56.        65.       149.       10.     460
25.        65.       162.       16.     470
37.        68.       152.       16.     480
46.        7C.       159.       15.     490
41.        69.       137.       14.     500
62.        71.       163.       12.     510
29.        72.       191.        4.     520
19.        68.       168.       10.     530
46.        63.       158.       16.     540
37.        64.       139.       18.     550
34.        68.       156.       10.     560
64.        67.       153.       12.     570
57.        67.       141.       13.     580
32.        68.       157.       17.     590
29.        7C.       183.       15.     600
53.        72.       164.       18.     610
47.        72.       156.       18.     620
56.        73.       16C.       16.     630
61.        74.       169.       12.     640
21.        68.       161.       10.     650
25.        76.       178.       11.     660
23.        72.       157.       16.     670
29.        68.       186.       16.     680
39.        7C.       159.       14.     690
42.        7C.       154.       1C.     700
56.        62.       159.       12.     710
63.        7C.       177.       12.     720
51.        71.       161.        9.     730
41.        66.       158.       10.     740
33.        69.       158.       16.     750
37.        68.       157.       16.     760
25.        7C.       163.       15.     770
63.        68.       159.       12.     780
53.        71.       202.        6.     790
51.        72.       167.       14.     800
47.        73.       164.       14.     810
39.        75.       151.       12.     820
28.        68.       166.       10.     830
64.        69.       156.       16.     840
55.        67.       144.       16.     850
51.        66.       177.       10.     860
46.        65.       157.       12.     870
72.        66.       125.       10.     880
66.        65.       131.       12.     890
28.        74.       149.       18.     900
27.        71.       168.       11.     910
23.        72.       158.       12.     920
23.        72.       163.       12.     930
60.        68.       157.        9.     940
3C.        66.       142.       10.     950
39.        67.       162.       16.     960
46.        74.       154.       16.     970
5C.        68.       158.       10.     980
61.        66.       161.       14.     990
36.        64.       157.       15.    1000
32.        71.       156.       16.    1010
           1         2         65     1020
           4         6          8     1030
      3   12C.       2C.       210.    1040
      1    2C.        7.        7C.    1050
      4    1C.       12.        2C.    1060
```

Figure 12.

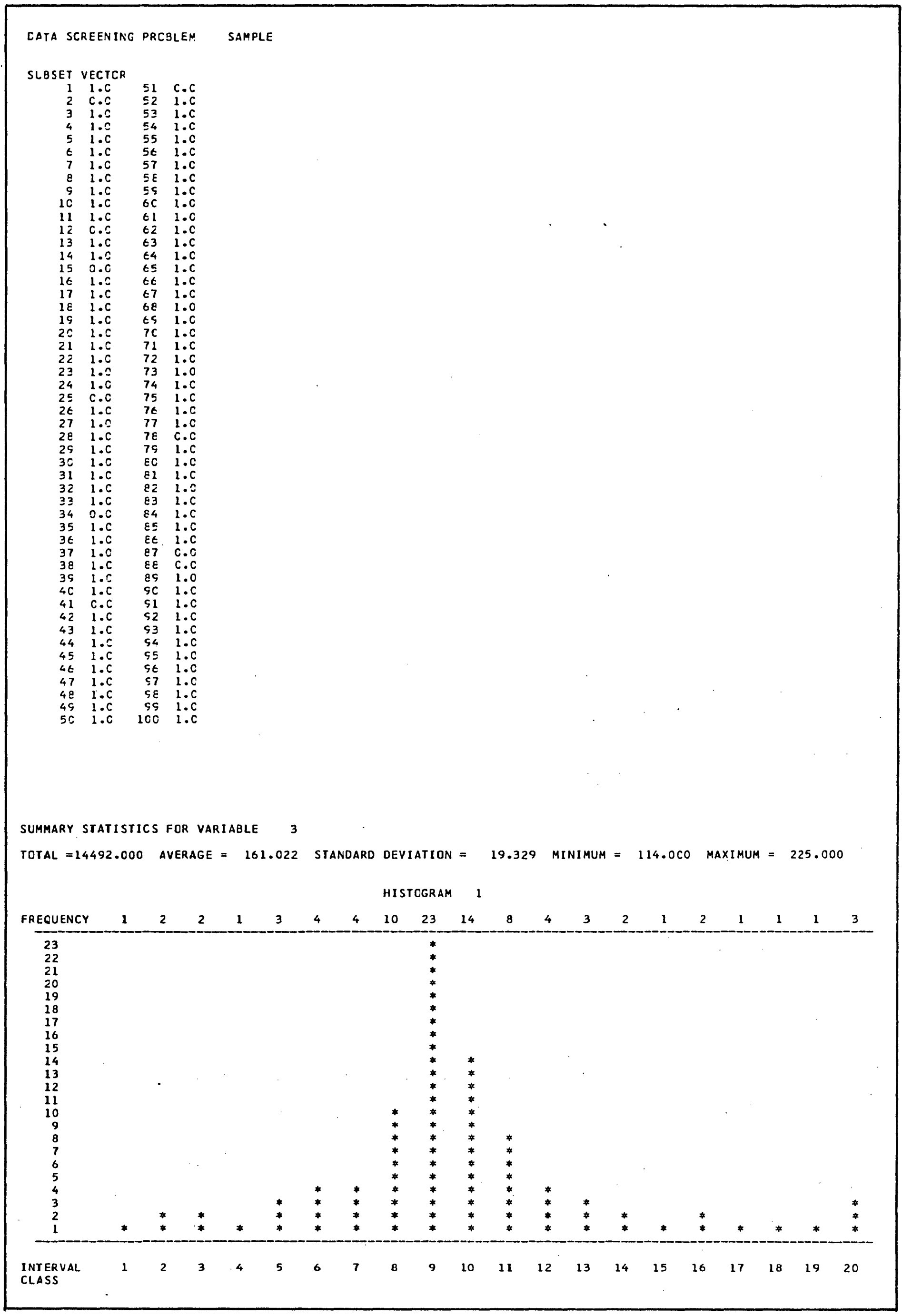
Output

Description

The output consists of the subset vector whose element values indicate which corresponding observations are rejected (element = zero) and accepted (element = nonzero), summary statistics for each selected variable, and a histogram of frequencies versus intervals for that variable.

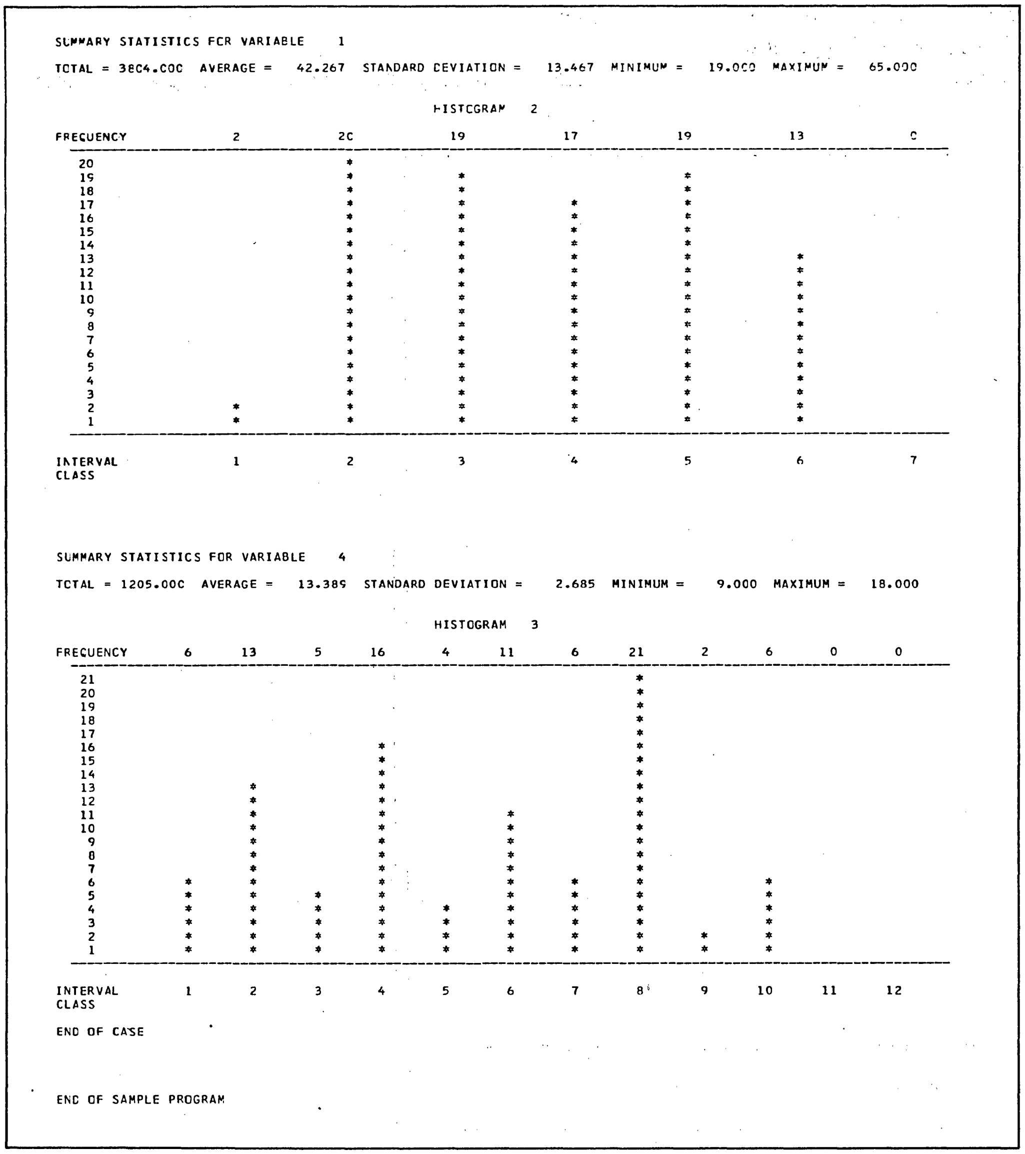
Sample

The output listing for the sample problem is shown in Figure 13.



```
 DATA SCREENING PROBLEM    SAMPLE

 SUBSET VECTOR
      1  1.0     51  0.0
      2  0.0     52  1.0
      3  1.0     53  1.0
      4  1.0     54  1.0
      5  1.0     55  1.0
      6  1.0     56  1.0
      7  1.0     57  1.0
      8  1.0     58  1.0
      9  1.0     59  1.0
     10  1.0     60  1.0
     11  1.0     61  1.0
     12  0.0     62  1.0
     13  1.0     63  1.0
     14  1.0     64  1.0
     15  0.0     65  1.0
     16  1.0     66  1.0
     17  1.0     67  1.0
     18  1.0     68  1.0
     19  1.0     69  1.0
     20  1.0     70  1.0
     21  1.0     71  1.0
     22  1.0     72  1.0
     23  1.0     73  1.0
     24  1.0     74  1.0
     25  0.0     75  1.0
     26  1.0     76  1.0
     27  1.0     77  1.0
     28  1.0     78  0.0
     29  1.0     79  1.0
     30  1.0     80  1.0
     31  1.0     81  1.0
     32  1.0     82  1.0
     33  1.0     83  1.0
     34  0.0     84  1.0
     35  1.0     85  1.0
     36  1.0     86  1.0
     37  1.0     87  0.0
     38  1.0     88  0.0
     39  1.0     89  1.0
     40  1.0     90  1.0
     41  0.0     91  1.0
     42  1.0     92  1.0
     43  1.0     93  1.0
     44  1.0     94  1.0
     45  1.0     95  1.0
     46  1.0     96  1.0
     47  1.0     97  1.0
     48  1.0     98  1.0
     49  1.0     99  1.0
     50  1.0    100  1.0



SUMMARY STATISTICS FOR VARIABLE    3

TOTAL =14492.000  AVERAGE =  161.022  STANDARD DEVIATION =    19.329  MINIMUM =  114.000  MAXIMUM =  225.000

                                               HISTOGRAM  1

FREQUENCY    1    2    2    1    3    4    4   10   23   14    8    4    3    2    1    2    1    1    1    3
-------------------------------------------------------------------------------------------------------------
   23                                                *
   22                                                *
   21                                                *
   20                                                *
   19                                                *
   18                                                *
   17                                                *
   16                                                *
   15                                                *
   14                                                *    *
   13                                                *    *
   12                                                *    *
   11                                                *    *
   10                                           *    *    *
    9                                           *    *    *
    8                                           *    *    *    *
    7                                           *    *    *    *
    6                                           *    *    *    *
    5                                           *    *    *    *
    4                                 *    *    *    *    *    *    *
    3                            *    *    *    *    *    *    *    *    *                              *
    2             *    *         *    *    *    *    *    *    *    *    *    *         *              *
    1        *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *
-------------------------------------------------------------------------------------------------------------

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

Figure 13.



```
SUMMARY STATISTICS FCR VARIABLE    1

TCTAL = 38C4.COC  AVERAGE =   42.267  STANDARD CEVIATION =   13.467  MINIMUM =   19.0C0  MAXIMUM =   65.000

                                              HISTCGRAM   2

FRECUENCY             2            2C            19            17            19            13             C
  -------------------------------------------------------------------------------------------------------------
  20                                *
  19                                *             *                           *
  18                                *             *                           *
  17                                *             *             *             *
  16                                *             *             *             *
  15                                *             *             *             *
  14                                *             *             *             *
  13                                *             *             *             *             *
  12                                *             *             *             *             *
  11                                *             *             *             *             *
  10                                *             *             *             *             *
   9                                *             *             *             *             *
   8                                *             *             *             *             *
   7                                *             *             *             *             *
   6                                *             *             *             *             *
   5                                *             *             *             *             *
   4                                *             *             *             *             *
   3                                *             *             *             *             *
   2                  *             *             *             *             *             *
   1                  *             *             *             *             *             *
  -------------------------------------------------------------------------------------------------------------

INTERVAL              1             2             3             4             5             6             7
CLASS



SUMMARY STATISTICS FOR VARIABLE    4

TCTAL = 1205.00C  AVERAGE =   13.389  STANDARD DEVIATION =    2.685  MINIMUM =    9.000  MAXIMUM =   18.000

                                              HISTOGRAM   3

FRECUENCY       6      13       5      16       4      11       6      21       2       6       0       0
  -------------------------------------------------------------------------------------------------------------
  21                                                                    *
  20                                                                    *
  19                                                                    *
  18                                                                    *
  17                                                                    *
  16                                    *                               *
  15                                    *                               *
  14                                    *                               *
  13                    *               *                               *
  12                    *               *                               *
  11                    *               *               *               *
  10                    *               *               *               *
   9                    *               *               *               *
   8                    *               *               *               *
   7                    *               *               *               *
   6            *       *               *               *       *       *               *
   5            *       *       *       *               *       *       *               *
   4            *       *       *       *       *       *       *       *               *
   3            *       *       *       *       *       *       *       *               *
   2            *       *       *       *       *       *       *       *       *       *
   1            *       *       *       *       *       *       *       *       *       *
  -------------------------------------------------------------------------------------------------------------

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

END OF CASE


END OF SAMPLE PROGRAM
```

Figure 13. (Continued)

Program Modifications

1. Changes in the input format statement of the special input routine, DAT1:

Only the format statement for input data may be changed. Since sample data are either two- or three-digit numbers, rather than using ten-column fields, as in the sample problem, each row of data might have been keypunched in three-column fields; if so, the format is changed to (7F(3, 0)). This format assumes seven 3-column fields per card.

2. If there are more than seven variables in a problem, each row of data is continued on the second card until the last data point is keypunched. However, each row must begin on a new card. If there is more than one data card per observation, the value of the data card count indicator (NCARD), which appears in columns 27-31 of the control card, must be changed to agree with the number of data cards.

3. Subroutine BOOL can be replaced if the user wishes to use a different boolean expression (see description in subroutine SBST). The boolean expression provided in the sample program is for both conditions to be satisfied:

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

Operating Instructions

The sample program for data screening is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT is used for output.

Timing

The execution time of this sample program on System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 40 seconds.

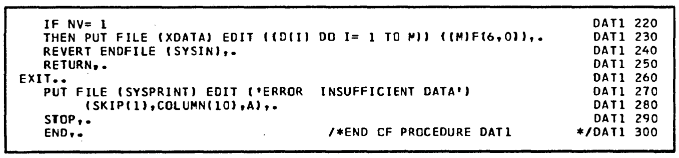
```
DACR..                                                                  DACR  10
/**********************************************************************/DACR  20
/*                                                                    */DACR  30
/*      TO PERFORM DATA SCREENING CALCULATIONS ON A SET OF            */DACR  40
/*      OBSERVATIONS.                                                 */DACR  50
/*                                                                    */DACR  60
/**********************************************************************/DACR  70
   PROCEDURE OPTIONS (MAIN),.                                           DACR  80
   DECLARE                                                              DACR  90
        (NO,NS,NN,NNN,NC,I,J,NOVAR,NX,NCOL,L1,L2)                       DACR 100
        FIXED BINARY,                                                   DACR 110
        PR1  CHARACTER (6),                                             DACR 120
        ERROR EXTERNAL CHARACTER(1),                                    DACR 130
        CH CHARACTER (80),                                              DACR 140
        (NV,NCARD) EXTERNAL,                                            DACR 150
        BOOL ENTRY,.                                                    DACR 160
/*                                                                    */DACR 170
        ON ENDFILE (SYSIN) GO TO EXIT,.                                 DACR 180
STRT..                                                                  DACR 190
   GET EDIT (CH) (A(80)),.                                              DACR 200
   GET STRING (CH) EDIT (PR1,NO,NX,NC,NS,NCARD) (A(6),5 F(5)),.         DACR 210
/*                                                                    */DACR 220
/*      PR1.......PROBLEM NUMBER (MAY BE ALPHAMERIC)                  */DACR 230
/*      NO........NUMBER OF OBSERVATIONS                              */DACR 240
/*      NX........NUMBER OF VARIABLES                                 */DACR 250
/*      NC........NUMBER OF CONDITIONS                                */DACR 260
/*      NS........NUMBER OF SELECTIONS                                */DACR 270
/*      NCARD.....NUMBER OF DATA CARDS PER OBSERVATION                */DACR 280
/*                                                                    */DACR 290
        NCARD=NCARD*80,.                                                DACR 300
ONE..                                                                   DACR 310
   BEGIN,.                                                              DACR 320
   DECLARE                                                              DACR 330
        (A(NO,NX),C(3,NC),UBO(3),S(NO),R(NC),STATS(5),D(NX),DD(3))      DACR 340
        FLOAT BINARY,.                                                  DACR 350
/*                                                                    */DACR 360
/*      IF INPUT DATA IS TO BE SAVED ON A DATA SET, INITIALIZE        */DACR 370
/*      NV=1. OTHERWISE NV=0.                                         */DACR 380
/*                                                                    */DACR 390
   NV    =0,.                                                           DACR 400
        DO  I = 1 TO NO,.                 /* READ IN DATA             */DACR 410
        CALL DAT1(NX,D),.                                               DACR 420
             DO J = 1 TO NX,.                                           DACR 430
             A(I,J)=D(J),.                                              DACR 440
             END,.                                                      DACR 450
        END,.                                                           DACR 460
   NCARD=80,.                                                           DACR 470
        DO  I = 1 TO NC,.                 /* READ IN CONDITIONS       */DACR 480
        NNN  =3,.                                                       DACR 490
        CALL DAT1 (NNN,DD),.                                            DACR 500
             DO J = 1 TO 3,.                                            DACR 510
             C(J,I)=DD(J),.                                             DACR 520
             END,.                                                      DACR 530
        END,.                                                           DACR 540
   CALL SBST (A,C,R,BOOL,S,NO,NX,NC),.                                  DACR 550
   PUT EDIT ('DATA SCREENING PROBLEM',PR1) (PAGE,COLUMN(10),A,X(4),A),.DACR 560
   IF ERROR NE '0'                                                      DACR 570
   THEN DO,.                                                            DACR 580
        PUT EDIT ('IN ROUTINE SBST ERROR CODE = ',ERROR)                DACR 590
             (SKIP(2),COLUMN(10),A,A(1)),.                              DACR 600
        GO TO FIN,.                                                     DACR 610
        END,.                                                           DACR 620
   PUT EDIT ('SUBSET VECTOR') (SKIP(3),COLUMN(10),A,SKIP(3)),.          DACR 630
   NCOL =CEIL(NO/50),.                                                  DACR 640
   IF NCOL LE 1                                                         DACR 650
   THEN PUT EDIT ((I,S(I) DO I= 1 TO NO)) (COLUMN(10),F(6),F(5,1)),.    DACR 660
   ELSE DO,.                                                            DACR 670
        L1    =0,.                                                      DACR 680
             DO I = 1 TO 50,.                                           DACR 690
             L1    =L1+1,.                                              DACR 700
             L2    =50*(NCOL-1)+L1,.                                    DACR 710
             IF L2= NO                                                  DACR 720
             THEN NCOL =NCOL-1,.                                        DACR 730
             PUT EDIT ((J,S(J) DO J= L1 TO L2 BY 50)) (SKIP,COLUMN(10) DACR 740
                  ,(9)(F(6),F(5,1))),.                                  DACR 750
             END,.                                                      DACR 760
        END,.                                                           DACR 770
        DO J = 1 TO NS,.                                                DACR 780
        GET EDIT (CH) (A(80)),.                                         DACR 790
        GET STRING (CH) EDIT (NOVAR,(UBO(I) DO I= 1 TO 3))              DACR 800
             (F(5),3 F(10,0)),.                                         DACR 810
/*                                                                    */DACR 820
/*      NOVAR.....NUMBER OF THE VARIABLE TO BE TABULATED              */DACR 830
/*      UBO(1)....LOWER BOUND                                         */DACR 840
/*      UBO(2)....NUMBER OF INTERVALS                                 */DACR 850
/*      UBO(3)....UPPER BOUND                                         */DACR 860
/*                                                                    */DACR 870
   NN    =UBO(2),.                                                      DACR 880
TWO..                                                                   DACR 890
   BEGIN,.                                                              DACR 900
        DECLARE                                                         DACR 910
             (FREQ(NN),PCT(NN)) FLOAT BINARY,.                          DACR 920
        CALL TAB1 (A,S,NOVAR,UBO,FREQ,PCT,STATS,NO,NX),.                DACR 930
        IF ERROR NE '0'                                                 DACR 940
        THEN PUT EDIT ('IN ROUTINE TAB1 ERROR CODE = ',ERROR)           DACR 950
             (SKIP(1),COLUMN(10),A,A(1)),.                              DACR 960
        ELSE DO,.                                                       DACR 970
             PUT EDIT ('SUMMARY STATISTICS FOR VARIABLE  ',NOVAR)       DACR 980
             (PAGE,SKIP(4),COLUMN(10),A,F(3)),.                         DACR 990
             PUT EDIT ('TOTAL =',STATS(1),'AVERAGE =',STATS(2),         DACR1000
                  'STANDARD DEVIATION =',STATS(3),'MINIMUM =',STATS(4),DACR1010
                  'MAXIMUM =',STATS(5))                                 DACR1020
                  (SKIP(2),COLUMN(10),5(A,F(9,3),X(2))),.               DACR1030
             CALL HIST (J,FREQ,NN),.                                    DACR1040
```

```
                  END,.                                                 DACR1050
             END,.                                                      DACR1060
        END,.                                                           DACR1070
        PUT EDIT ('END OF CASE') (SKIP(2),COLUMN(10),A),.               DACR1080
        END,.                                                           DACR1090
        GO TO STRT,.                                                    DACR1100
EXIT..                                                                  DACR1110
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')                   DACR1120
        (SKIP(5),COLUMN(10),A),.                                        DACR1130
FIN..                                                                   DACR1140
   END,.                                  /*END OF PROCEDURE DACR     */DACR1150
```

```
BOOL..                                                                  BOOL  10
/**********************************************************************/BOCL  20
/*                                                                    */BOOL  30
/*      TO PERFCRM A BOOLEAN CPERATICN FCR THE PROCEDURE SBST, WHICH  */BOCL  40
/*      IS USED BY THE DATA SCREENING SAMPLE PRCGRAM.                 */BOOL  50
/*                                                                    */BOCL  60
/**********************************************************************/BOCL  70
   PROCEDURE (R,T),.                                                    BOOL  80
   DECLARE (R(*),T) FLCAT BINARY,.                                      BOCL  9C
/*    BOOL CHECKS CNLY THE FIRST TWC CONDITICNS CF PROCEDURE SBST     */BOCL 100
   T    =R(1)*R(2),.                                                    BOOL 110
   RETURN,.                                                             BOCL 120
   END,.                                  /*END CF PROCEDURE BOOL     */BOOL 130
```

```
HIST..                                                                  HIST  10
/**********************************************************************/HIST  20
/*                                                                    */HIST  30
/*      TC PLOT A HISTOGRAM CF FREQUENCIES FCR THE DATA SCREENING     */HIST  40
/*      SAMPLE PROGRAM.                                               */HIST  50
/*                                                                    */HIST  60
/**********************************************************************/HIST  70
   PROCEDURE (NZ,FREQ,IN),.                                             HIST  80
   DECLARE                                                              HIST  90
        (I,IN,IX,J,JSCAL,L,MAX,NU,NZ)                                   HIST 100
        FIXED BINARY,                                                   HIST 110
        (K,JOUT(IN)) CHARACTER (1),                                     HIST 120
        (FREQ(*),FMAX,X) FLOAT BINARY,.                                 HIST 130
/*                                                                    */HIST 140
/*      PRINT TITLE AND FREQUENCY VECTOR                              */HIST 150
/*                                                                    */HIST 160
   PUT EDIT ('HISTOGRAM ',NZ) (SKIP(3),COLUMN(57),A,F(3)),.             HIST 170
   NU    =FLOOR(100/IN),.                                               HIST 180
   PUT EDIT ('FREQUENCY',(FREQ(I) DO I = 1 TO IN))                      HIST 190
        (SKIP(2),COLUMN(10),A,(IN)F(NU)),.                              HIST 200
   PUT EDIT ('-----------------------------------------------',        HIST 210
        '-------------------------------------------------')           HIST 220
        (R(FM1)),.                                                      HIST 230
FM1..                                                                   HIST 240
   FORMAT (SKIP,COLUMN(12),A,A),.                                       HIST 250
   FMAX =0,.                              /* FIND LARGEST FREQUENCY   */HIST 260
        DO I = 1 TO IN,.                                                HIST 270
        IF FREQ(I) GT FMAX                                              HIST 280
        THEN FMAX =FREQ(I),.                                            HIST 290
        END,.                                                           HIST 300
   JSCAL=1,.                              /* SCALE IF NECESSARY       */HIST 310
   IF FMAX GT 50                                                        HIST 320
   THEN DO,.                                                            HIST 330
        JSCAL=FLOOR((FMAX+49)/50),.                                     HIST 340
        PUT EDIT ('EACH','*',' EQUAL ',JSCAL,' POINTS')                 HIST 350
             (SKIP,COLUMN(10),A,A(1),A,F(2),A,SKIP),.                   HIST 360
        END,.                                                           HIST 370
   JOUT =' ',.                            /* CLEAR CUTPUT AREA TO BLANKS */HIST 380
/*                                                                    */HIST 390
/*      LOCATE FREQUENCIES IN EACH INTERVAL                           */HIST 400
/*                                                                    */HIST 410
   MAX  =FLOOR(FMAX/JSCAL),.                                            HIST 420
        DO I = 1 TO MAX,.                                               HIST 430
        X    =MAX-(I-1),.                                               HIST 440
             DO J = 1 TO IN,.                                           HIST 450
             IF FREQ(J)/JSCAL GE X                                      HIST 460
             THEN JOUT(J)='*',.                                         HIST 470
             END,.                                                      HIST 480
        IX   =X*JSCAL,.                                                 HIST 490
/*                                                                    */HIST 500
/*      PRINT LINE OF FREQUENCIES                                     */HIST 510
/*                                                                    */HIST 520
        PUT EDIT (IX,(JOUT(L) DC L = 1 TO IN)) (SKIP,COLUMN(10),F(5),  HIST 530
             X(4),(IN)(X(NU-1),A(1))),.                                 HIST 540
        END,.                                                           HIST 550
        CO I = 1 TO IN,.                  /* GENERATE CCNSTANTS       */HIST 560
        FREQ(I)=I,.                                                     HIST 570
        END,.                                                           HIST 580
   PUT EDIT ('-----------------------------------------------',        HIST 590
        '-------------------------------------------------')           HIST 600
        (R(FM1)),.                                                      HIST 610
   PUT EDIT ('INTERVAL ',(FREQ(I) DO I = 1 TC IN))                      HIST 62C
        (SKIP(2),CCLUMN(1C),A,(IN)F(NU)),.                              HIST 63C
   PUT EDIT ('CLASS') (SKIP,COLUMN(10),A),.                             HIST 640
   RETURN,.                                                             HIST 650
   END,.                                  /*END CF PROCEDURE HIST     */HIST 66C
```

```
DAT1..                                                                  DAT1  10
/**********************************************************************/DAT1  20
/*                                                                    */DAT1  30
/*      TO READ FLCATING PCINT DATA, ONE OBSERVATION AT A TIME.       */DAT1  40
/*      DATA MAY BE SAVED ON A DATA SET.                              */DAT1  50
/*                                                                    */DAT1  60
/**********************************************************************/DAT1  70
   PROCEDURE (M,D),.                                                    DAT1  80
   DECLARE                                                              DAT1  9C
        XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),        DAT1 100
        (NCARD,NV) EXTERNAL,                                            DAT1 110
        CH CHARACTER(NCARD),                                            DAT1 120
        (I,M,MM) BINARY FIXED,                                          DAT1 130
        D(*) FLOAT BINARY,.                                             DAT1 14C
/*                                                                    */DAT1 150
   ON ENDFILE (SYSIN)                                                   DAT1 160
   GO TO EXIT,.                                                         DAT1 170
   GET EDIT (CH) (A(NCARD)),.                                           DAT1 180
   MM   =CEIL(M/7),.                                                    DAT1 190
   GET STRING (CH) EDIT ((D(I) DO I= 1 TC M))                           DAT1 200
        ((MM)((7)F(10,0),X(10))),.                                      DAT1 210
```



```
   IF NV= 1                                                            DAT1 220
   THEN PUT FILE (XDATA) EDIT ((D(I) DO I= 1 TO M)) ((M)F(6,0)),.      DAT1 230
   REVERT ENDFILE (SYSIN),.                                            DAT1 240
   RETURN,.                                                            DAT1 250
EXIT..                                                                 DAT1 260
   PUT FILE (SYSPRINT) EDIT ('ERROR  INSUFFICIENT DATA')               DAT1 270
        (SKIP(1),COLUMN(10),A),.                                       DAT1 280
   STOP,.                                                              DAT1 290
   END,.                          /*END CF PROCEDURE DAT1             */DAT1 300
```

MULTIPLE LINEAR REGRESSION REGR

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 1. The first five variables are independent variables (predictors), and the last is the dependent variable (criteria). All five independent variables are used to predict the dependent variable in the first analysis, and only the second, third, and fifth variables are used to predict the dependent variable in the second analysis.

Table 1. Sample Data for Multiple Linear Regression

	Variables					
Observation	X_1	X_2	X_3	X_4	X_5	X_6
1	29	289	216	85	14	1
2	30	391	244	92	16	2
3	30	424	246	90	18	2
4	30	313	239	91	10	0
5	35	243	275	95	30	2
6	35	365	219	95	21	2
7	43	396	267	100	39	3
8	43	356	274	79	19	2
9	44	346	255	126	56	3
10	44	156	258	95	28	0
11	44	278	249	110	42	4
12	44	349	252	88	21	1
13	44	141	236	129	56	1
14	44	245	236	97	24	1
15	45	297	256	111	45	3
16	45	310	262	94	20	2
17	45	151	339	96	35	3
18	45	370	357	88	15	4
19	45	379	198	147	64	4
20	45	463	206	105	31	3
21	45	316	245	132	60	4
22	45	280	225	108	36	4
23	44	395	215	101	27	1
24	49	139	220	136	59	0
25	49	245	205	113	37	4
26	49	373	215	88	25	1
27	51	224	215	118	54	3
28	51	677	210	116	33	4
29	51	424	210	140	59	4
30	51	150	210	105	30	0

Program

Description

The multiple linear regression program consists of the main program named REGR, two special input routines named DAT2 and IDT1, and four subroutines from the Scientific Subroutine Package: CORR, ORDR, MINV, and MLTR.

Capacity

1. Up to 99,999 observations can be read if observations are read into the computer one at a time by the special input subroutine named DAT2. If all data are to be stored in core before the calculation of correlation coefficients, the limitation on the number of observations depends on the size of core storage available for input data.
2. Up to 96 variables can be handled.
3. Up to 99 selections can be handled.
4. Up to eight cards per observation can be read.
5. (12 F (6, 0)) format for input data cards.

Therefore, if a problem satisfies the above conditions, the sample program need not be modified. If the input data cards are prepared using a different format, the input format in the subroutine DAT2 must be modified. The general rules for program modifications are described later.

6. Up to 40 independent variables for one selection can be read.

Input

Control Cards

One control card is required for each problem and is read by the main program, REGR. This card is prepared as follows:

Columns	Contents	For Sample Problem
1-6	Problem number (may be alphameric)	SAMPLE
7-11	Number of observations	00030
12-13	Number of variables	06
14-15	Number of selections (see below)	02
16-17	Number of data cards per observation	01

Leading zeros do not have to be keypunched.

Data Cards

Since input data is read into the computer one observation at a time, each row of data in Table 1 is keypunched on a separate card using the format (12 F (6, 0)). This format assumes twelve 6-column fields per card.

Selection Cards

For each selection there must be at least two cards, as described below. If the number of selections specified is zero, the program will terminate. An error message is printed out.

The first card is used to specify a single dependent variable in a multiple linear regression analysis. Any one variable in the set of original variables can be designated as a dependent variable, and any positive number of variables can be specified as independent variables. Selection of a single dependent variable and a set of independent variables can be performed over and over again using the same set of original variables.

The first card is prepared as follows:

		For Sample Problem	
Columns	Contents	Selection 1	Selection 2
1-2	Option code for table of residuals 0 if table is not desired; 1 if table 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, (the subscript numbers of individual variables are specified below).	05	03

The second card is prepared as follows:

		For Sample Problem	
Columns	Contents	Selection 1	Selection 2
1-2	1st independent variable included	01	02
3-4	2nd independent variable included	02	03
5-6	3rd independent variable included	03	05
7-8	4th independent variable included	04	
9-10	5th independent variable included	05	
etc.			

The input format of (40 F (2)) is used for the second card.

Deck Setup

Deck setup is shown in Figure 14.

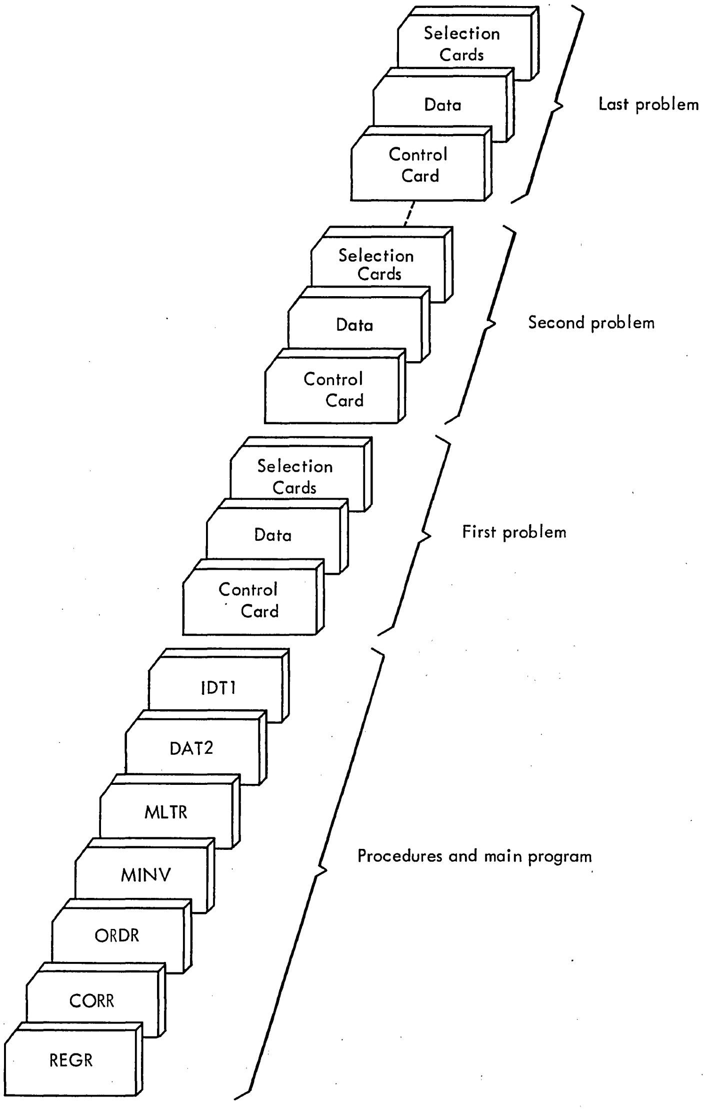
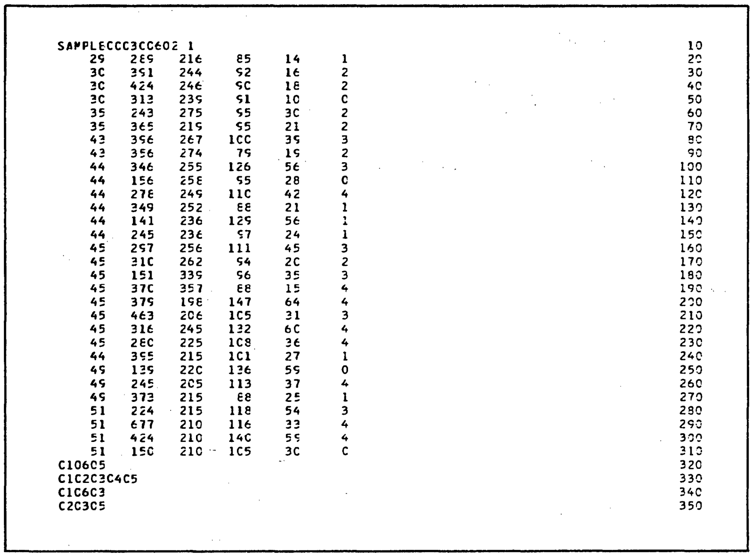


Figure 14.

Sample

The listing of input cards for the sample problem is shown in Figure 15.



```
SAMPLE000300602 1                                                        10
    29   289   216    85    14     1                                     20
    30   391   244    92    16     2                                     30
    30   424   246    90    18     2                                     40
    30   313   239    91    10     0                                     50
    35   243   275    95    30     2                                     60
    35   365   219    95    21     2                                     70
    43   396   267   100    39     3                                     80
    43   356   274    79    19     2                                     90
    44   346   255   126    56     3                                    100
    44   156   258    95    28     0                                    110
    44   278   249   110    42     4                                    120
    44   349   252    88    21     1                                    130
    44   141   236   129    56     1                                    140
    44   245   236    97    24     1                                    150
    45   297   256   111    45     3                                    160
    45   310   262    94    20     2                                    170
    45   151   339    96    35     3                                    180
    45   370   357    88    15     4                                    190
    45   379   198   147    64     4                                    200
    45   463   206   105    31     3                                    210
    45   316   245   132    60     4                                    220
    45   280   225   108    36     4                                    230
    44   395   215   101    27     1                                    240
    49   139   220   136    59     0                                    250
    49   245   205   113    37     4                                    260
    49   373   215    88    25     1                                    270
    51   224   215   118    54     3                                    280
    51   677   210   116    33     4                                    290
    51   424   210   140    59     4                                    300
    51   150   210   105    30     0                                    310
010605                                                                  320
0102030405                                                              330
010603                                                                  340
020305                                                                  350
```

Figure 15.

Output

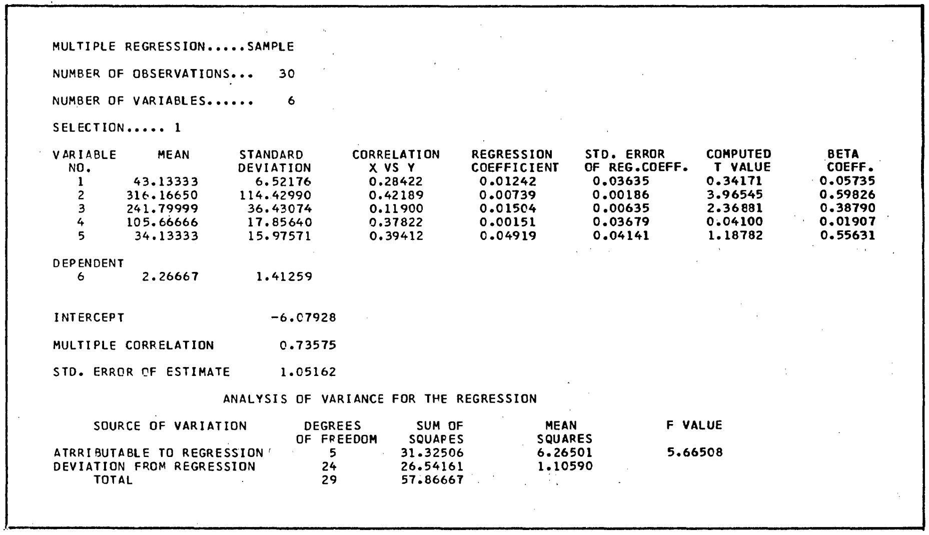
Description

The output based on the selection card of the sample program for multiple linear regression includes:

1. Means
2. Standard deviations
3. Correlation coefficients between independent variables and dependent variables
4. Regression coefficients
5. Standard errors of regression coefficients
6. Computed T values
7. Intercept
8. Multiple correlation coefficients
9. Standard error of estimate
10. Beta coefficients
11. Analysis of variance for the multiple regression
12. Table of residuals (optional)

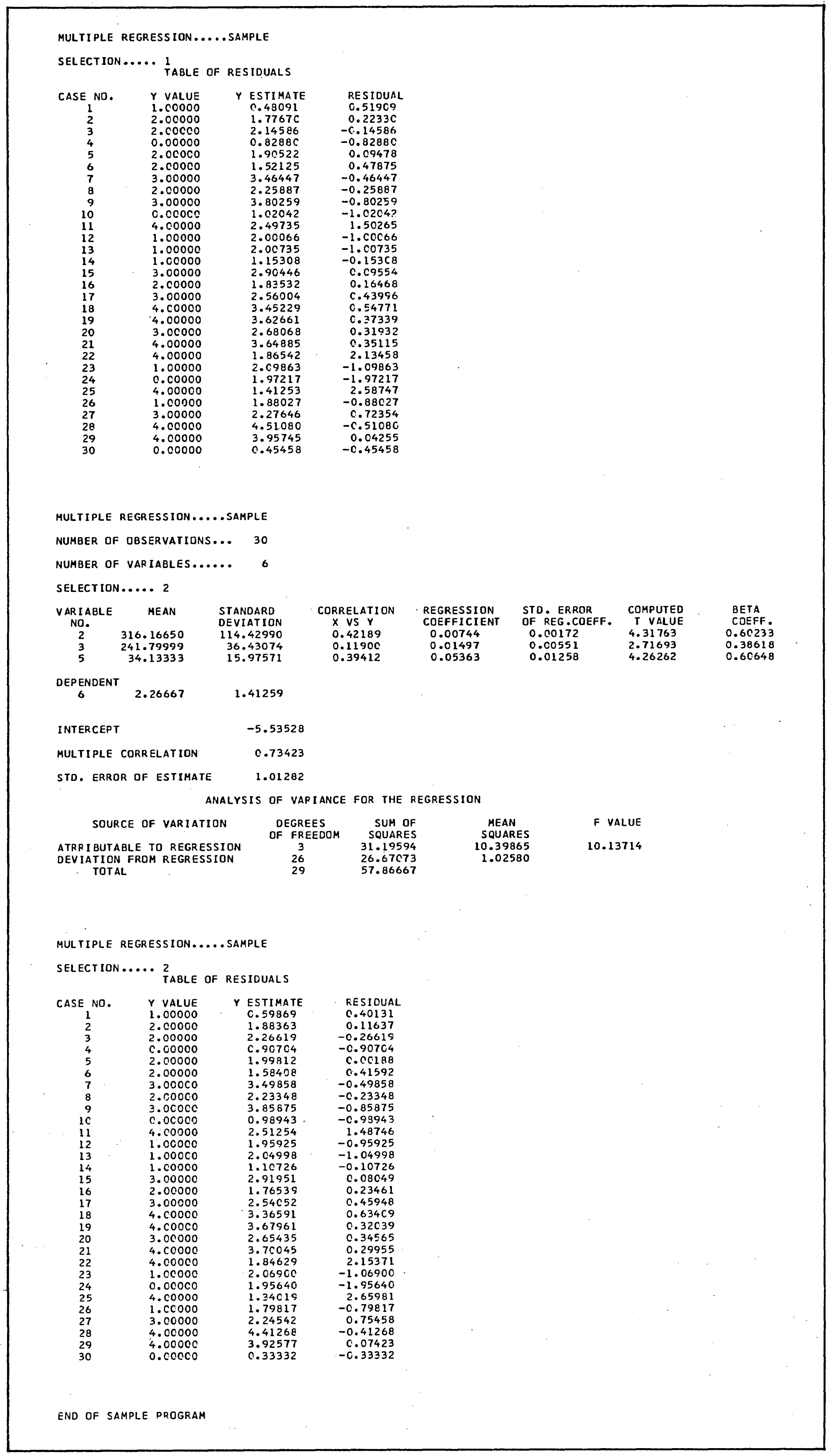
Sample

The output listing for the sample problem is shown in Figure 16.



```
MULTIPLE REGRESSION.....SAMPLE

NUMBER OF OBSERVATIONS...   30

NUMBER OF VARIABLES......    6

SELECTION..... 1

VARIABLE     MEAN      STANDARD       CORRELATION     REGRESSION    STD. ERROR     COMPUTED       BETA
  NO.                  DEVIATION        X VS Y        COEFFICIENT   OF REG.COEFF.   T VALUE       COEFF.
   1       43.13333      6.52176        0.28422        0.01242        0.03635       0.34171      0.05735
   2      316.16650    114.42990        0.42189        0.00739        0.00186       3.96545      0.59826
   3      241.79999     36.43074        0.11900        0.01504        0.00635       2.36881      0.38790
   4      105.66666     17.85640        0.37822        0.00151        0.03679       0.04100      0.01907
   5       34.13333     15.97571        0.39412        0.04919        0.04141       1.18782      0.55631

DEPENDENT
   6        2.26667      1.41259


INTERCEPT                   -6.07928

MULTIPLE CORRELATION         0.73575

STD. ERROR OF ESTIMATE       1.05162

                   ANALYSIS OF VARIANCE FOR THE REGRESSION

     SOURCE OF VARIATION        DEGREES      SUM OF           MEAN             F VALUE
                               OF FREEDOM    SQUAPES          SQUARES
ATRRIBUTABLE TO REGRESSION         5        31.32506         6.26501          5.66508
DEVIATION FROM REGRESSION         24        26.54161         1.10590
     TOTAL                        29        57.86667
```

Figure 16



```
MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 1
                 TABLE OF RESIDUALS

CASE NO.     Y VALUE      Y ESTIMATE      RESIDUAL
    1        1.00000        0.48091        0.51909
    2        2.00000        1.77670        0.22330
    3        2.00000        2.14586       -0.14586
    4        0.00000        0.82880       -0.82880
    5        2.00000        1.90522        0.09478
    6        2.00000        1.52125        0.47875
    7        3.00000        3.46447       -0.46447
    8        2.00000        2.25887       -0.25887
    9        3.00000        3.80259       -0.80259
   10        0.00000        1.02042       -1.02042
   11        4.00000        2.49735        1.50265
   12        1.00000        2.00066       -1.00066
   13        1.00000        2.00735       -1.00735
   14        1.00000        1.15308       -0.15308
   15        3.00000        2.90446        0.09554
   16        2.00000        1.83532        0.16468
   17        3.00000        2.56004        0.43996
   18        4.00000        3.45229        0.54771
   19        4.00000        3.62661        0.37339
   20        3.00000        2.68068        0.31932
   21        4.00000        3.64885        0.35115
   22        4.00000        1.86542        2.13458
   23        1.00000        2.09863       -1.09863
   24        0.00000        1.97217       -1.97217
   25        4.00000        1.41253        2.58747
   26        1.00000        1.88027       -0.88027
   27        3.00000        2.27646        0.72354
   28        4.00000        4.51080       -0.51080
   29        4.00000        3.95745        0.04255
   30        0.00000        0.45458       -0.45458



MULTIPLE REGRESSION.....SAMPLE

NUMBER OF OBSERVATIONS...   30

NUMBER OF VARIABLES......    6

SELECTION..... 2

VARIABLE      MEAN        STANDARD      CORRELATION    REGRESSION      STD. ERROR      COMPUTED         BETA
  NO.                     DEVIATION       X VS Y       COEFFICIENT    OF REG.COEFF.    T VALUE          COEFF.
   2       316.16650     114.42990       0.42189        0.00744         0.00172        4.31763         0.60233
   3       241.79999      36.43074       0.11900        0.01497         0.00551        2.71693         0.38618
   5        34.13333      15.97571       0.39412        0.05363         0.01258        4.26262         0.60648

DEPENDENT
   6         2.26667       1.41259


INTERCEPT                   -5.53528

MULTIPLE CORRELATION         0.73423

STD. ERROR OF ESTIMATE       1.01282
                       ANALYSIS OF VARIANCE FOR THE REGRESSION

     SOURCE OF VARIATION          DEGREES       SUM OF            MEAN             F VALUE
                                 OF FREEDOM     SQUARES          SQUARES
ATRRIBUTABLE TO REGRESSION           3         31.19594         10.39865          10.13714
DEVIATION FROM REGRESSION           26         26.67073          1.02580
     TOTAL                          29         57.86667



MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 2
                 TABLE OF RESIDUALS

CASE NO.     Y VALUE      Y ESTIMATE      RESIDUAL
    1        1.00000        0.59869        0.40131
    2        2.00000        1.88363        0.11637
    3        2.00000        2.26619       -0.26619
    4        0.00000        0.90704       -0.90704
    5        2.00000        1.99812        0.00188
    6        2.00000        1.58408        0.41592
    7        3.00000        3.49858       -0.49858
    8        2.00000        2.23348       -0.23348
    9        3.00000        3.85875       -0.85875
   10        0.00000        0.98943       -0.98943
   11        4.00000        2.51254        1.48746
   12        1.00000        1.95925       -0.95925
   13        1.00000        2.04998       -1.04998
   14        1.00000        1.10726       -0.10726
   15        3.00000        2.91951        0.08049
   16        2.00000        1.76539        0.23461
   17        3.00000        2.54052        0.45948
   18        4.00000        3.36591        0.63409
   19        4.00000        3.67961        0.32039
   20        3.00000        2.65435        0.34565
   21        4.00000        3.70045        0.29955
   22        4.00000        1.84629        2.15371
   23        1.00000        2.06900       -1.06900
   24        0.00000        1.95640       -1.95640
   25        4.00000        1.34019        2.65981
   26        1.00000        1.79817       -0.79817
   27        3.00000        2.24542        0.75458
   28        4.00000        4.41268       -0.41268
   29        4.00000        3.92577        0.07423
   30        0.00000        0.33332       -0.33332


END OF SAMPLE PROGRAM
```

Figure 16. (Continued)

Program Modifications

Input data in a different format can also be handled by providing a special format statement.

1. Changes in the input format statement of the special input routine DAT2:

    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 might have been keypunched in six 3-column fields; if so, the format is changed to (6 F (3, 0)).

    The special input subroutine, DAT2, is normally written by the user to handle different formats for different problems. The user may modify this routine to perform listing of input data, transformation of data, and so on. When doing so, attention should be paid to the format statement in DAT2 (DAT2 230) which writes on the intermediate data set. The format in this statement must be the same as the format in statement REGR 1860.

2. If there are more than twelve variables in a problem, each row of data is continued on the next cards, until the last data point is keypunched. However, each row of data must begin on a new card.

    In the sample problem there is one data card per row, so the value of the card count indicator (NCARD), which appears in columns 16 and 17 of the control card, is set to one. If there is more than one data card per row, the value of the card count indicator (NCARD) must agree with the number of data cards per row.

3. Although the program will allow 96 variables, the maximum number of independent variables that may be specified on one selection is 40.

Error Messages

The following error conditions will result in messages:

1. The number of selections is not specified on the control card: NUMBER OF SELECTIONS NOT SPECIFIED. JOB TERMINATED.

Operating Instructions

The sample program for multiple linear regression is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output. A scratch tape (data set XDATA) is used as intermediate storage.

Timing

The execution time of this sample program on a System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 40 seconds.

```
REGR..                                                                  REGR  10
/**********************************************************************/REGR  20
/*                                                                    */REGR  30
/*      TO READ THE PROBLEM PARAMETER CARD FOR A MULTIPLE REGRESSION,*/REGR  40
/*      READ SUBSET SELECTION CARDS, CALL THE PROCEDURES TO CALCULATE*/REGR  50
/*      MEANS, STANDARD DEVIATIONS, SIMPLE AND MULTIPLE CORRELATION   */REGR  60
/*      COEFFICIENTS, REGRESSION COEFFICIENTS, T-VALUES, BETA COEFF-  */REGR  70
/*      ICIENTS, AND ANALYSIS OF VARIANCE FOR MULTIPLE REGRESSION,    */REGR  80
/*      AND PRINT THE RESULTS.                                        */REGR  90
/*                                                                    */REGR 100
/**********************************************************************/REGR 110
   PROCEDURE OPTIONS (MAIN),.                                           REGR 120
   DECLARE                                                              REGR 130
        (I,II,IO,J,K,L,M,MM,N,NDEP,NRESI,NS,L1,L2) FIXED BINARY,        REGR 140
        XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),        REGR 150
        (NCARD,NV) EXTERNAL,                                            REGR 160
        ERROR EXTERNAL CHARACTER (1),                                   REGR 170
        CH  CHARACTER (80),                                             REGR 180
        PR1 CHARACTER (6),.                                             REGR 190
/*                                                                    */REGR 200
FM1..                                                                   REGR 210
        FORMAT (A(6),F(5),3 F(2)),.                                     REGR 220
   ON ENDFILE (SYSIN) GO TO EXIT,.                                      REGR 230
/*                                                                    */REGR 240
/*      INPUT DATA IS SAVED IF NV IS SET TO 1                         */REGR 250
/*                                                                    */REGR 260
   NV   =1,.                                                            REGR 270
/*                                                                    */REGR 280
S100..                                                                  REGR 290
   GET EDIT (CH) (A(80)),.                                              REGR 300
   GET STRING (CH) EDIT (PR1,N,M,NS,NCARD) (R(FM1)),.                   REGR 310
/*                                                                    */REGR 320
/*      NAME - PROBLEM NUMBER (MAY BE ALPHAMERIC)                     */REGR 330
/*      N    - NUMBER OF OBSERVATIONS                                 */REGR 340
/*      M    - NUMBER OF VARIABLES                                    */REGR 350
/*      NS   - NUMBER OF SELECTIONS                                   */REGR 360
/*      NCARD- NUMBER OF DATA CARDS PER OBSERVATION                   */REGR 370
/*                                                                    */REGR 380
   NCARD=NCARD*80,.                                                     REGR 390
/*                                                                    */REGR 400
STRT..                                                                  REGR 410
   BEGIN,.                                                              REGR 420
FM2..                                                                   REGR 430
   FORMAT (PAGE,SKIP(4),COLUMN(10),A,A(6),SKIP(2),COLUMN(10),A,A,       REGR 440
        F(5),SKIP(2),COLUMN(10),A,F(5),SKIP(2),COLUMN(10),A,F(2)),.     REGR 450
   DECLARE                                                              REGR 460
        (X(1,1),W(M),RESI)                                              REGR 470
        FLOAT BINARY,                                                   REGR 480
        (R(M,M),RX(M,M),XBAR(M),RY(M),D(M),STD(M),ANS(10),FSUM,DET,CON)REGR 490
        BINARY FLOAT,.                 /*SINGLE PRECISION VERSION  /*S*/REGR 500
/*      BINARY FLOAT (53),.            /*DOUBLE PRECISION VERSION  /*D*/REGR 510
/*                                                                    */REGR 520
   IO   =0,.                                                            REGR 530
   X    =0,.                                                            REGR 540
   OPEN FILE (XDATA) OUTPUT,.                                           REGR 550
   CALL CORR (N,M,IO,X,XBAR,STD,RX,R,D),.                               REGR 560
   CLOSE FILE (XDATA),.                                                 REGR 570
   IF ERROR NE '0'                                                      REGR 580
   THEN PUT EDIT ('IN ROUTINE CORR ERROR CODE = ',ERROR)                REGR 590
        (SKIP(2),COLUMN(10),A,A(1)),.                                   REGR 600
/*                                                                    */REGR 610
/*      TEST NUMBER OF SELECTIONS                                     */REGR 620
/*                                                                    */REGR 630
   IF NS LE 0                                                           REGR 640
   THEN DO,.                                                            REGR 650
        PUT EDIT ('NUMBER OF SELECTIONS NOT SPECIFIED. JOB TERMINATED')REGR 660
            (SKIP(4),COLUMN(10),A),.                                    REGR 670
        GO TO S300,.                                                    REGR 680
        END,.                                                           REGR 690
        DO I = 1 TO NS,.                                                REGR 700
        PUT EDIT ('MULTIPLE REGRESSION.....',PR1,'NUMBER OF OBSERVA',   REGR 710
            'TIONS...',N,'NUMBER OF VARIABLES......',M,                 REGR 720
            'SELECTION.....',I) (R(FM2)),.                              REGR 730
/*                                                                    */REGR 740
/*      READ SUBSET SELECTION CARD                                    */REGR 750
/*                                                                    */REGR 760
   GET EDIT (CH) (A(80)),.                                              REGR 770
        GET STRING (CH) EDIT (NRESI,NDEP,K) (3 F(2)),.                  REGR 780
KRED..                                                                  REGR 790
   BEGIN,.                                                              REGR 800
FM3..                                                                   REGR 810
        FORMAT (SKIP,COLUMN(10),F(4),7 F(14,5)),.                       REGR 820
FM4..                                                                   REGR 830
        FORMAT (PAGE,SKIP(4),COLUMN(10),A,A(6),SKIP(2),COLUMN(10),      REGR 840
            A,F(2)),.                                                   REGR 850
        DECLARE                                                         REGR 860
            (RZ(K,K),B(K),SB(K),T(K),BETA(K),RT(K))                     REGR 870
            BINARY FLOAT,              /*SINGLE PRECISION VERSION  /*S*/REGR 880
/*          BINARY FLOAT (53),         /*DOUBLE PRECISION VERSION  /*D*/REGR 890
            (ISAVE(K+1))                                                REGR 900
            FIXED BINARY,.                                              REGR 910
/*                                                                    */REGR 920
        CALL IDT1 (K,ISAVE),.                                           REGR 930
/*                                                                    */REGR 940
/*      NRESI - OPTION CODE FOR TABLE OF RESIDUALS                    */REGR 950
/*              0 IF IT IS NOT DESIRED.                               */REGR 960
/*              1 IF IT IS DESIRED.                                   */REGR 970
/*      NDEP  - DEPENDENT VARIABLE.                                   */REGR 980
/*      K     - NUMBER OF INDEPENDENT VARIABLES INCLUDED              */REGR 990
/*      ISAVE - A VECTOR CONTAINING THE INDEPENDENT VARIABLES         */REGR1000
/*              INCLUDED                                              */REGR1010
/*                                                                    */REGR1020
        CALL ORDR (M,R,NDEP,K,ISAVE,RZ,RT),.                            REGR1030
        IF ERROR NE '0'                                                 REGR1040
        THEN DO,.                                                       REGR1050
            PUT EDIT ('IN ROUTINE ORDR ERROR CODE = ',ERROR)            REGR1060
                (SKIP(2),COLUMN(10),A,A(1)),.                           REGR1070
            GO TO S200,.                                                REGR1080
            END,.                                                       REGR1090
        CON  =0.0,.                                                     REGR1100
        CALL MINV(RZ,K,DET,CON),.                                       REGR1110
/*                                                                    */REGR1120
/*      TEST SINGULARITY OF THE MATRIX INVERTED                       */REGR1130
/*                                                                    */REGR1140
        IF ERROR NE '0'                                                 REGR1150
        THEN DO,.                                                       REGR1160
            PUT EDIT('IN ROUTINE MINV ERROR = ',ERROR) (SKIP(2),        REGR1170
                COLUMN(10),A,A(1)),.                                    REGR1180
            GO TO S200,.                                                REGR1190
            END,.                                                       REGR1200
/*                                                                    */REGR1210
```

```
         CALL MLTR (N,K,XBAR,STD,D,RZ,RT,ISAVE,B,SB,T,BETA,ANS),.
         IF ERROR NE '0'
         THEN DO,.
              PUT EDIT ('IN ROUTINE MLTR ERROR CODE = ',ERROR)
                   (SKIP(2),COLUMN(10),A,A(1)),.
              GO TO S200,.
              END,.
/*                                                                     */
/*       PRINT MEANS, STANDARD DEVIATIONS, INTERCORRELATIONS BETWEEN   */
/*       X AND Y, REGRESSION COEFFICIENTS, STANDARD DEVIATIONS OF      */
/*       REGRESSION COEFFICIENTS, COMPUTED T VALUES, AND BETA          */
/*       COEFFICIENTS.                                                 */
/*                                                                     */
         MM   =K+1,.
         PUT EDIT ('VARIABLE','MEAN','STANDARD','CORRELATION',
              'REGRESSION','STD. ERROR','COMPUTED','BETA','NO.',
              'DEVIATION','X VS Y','COEFFICIENT','OF REG.COEFF.',
              'T VALUE','COEFF.') (SKIP(2),COLUMN(10),A,X(5),A,
              X(6),A,X(6),A,X(4),A,X(4),A,X(5),A,X(7),A,SKIP,
              COLUMN(12),A,X(18),A,X(7),A,X(7),A,X(3),A,X(3),A,
              X(7),A),.
              DO J = 1 TO K,.
              L    =ISAVE(J),.
              PUT EDIT (L,XBAR(L),STD(L),RT(J),B(J),SB(J),T(J),BETA(J))
                   (R(FM3)),.
              END,.
         PUT EDIT ('DEPENDENT') (SKIP(2),COLUMN(10),A),.
         L    =ISAVE(MM),.
         PUT EDIT (L,XBAR(L),STD(L)) (R(FM3)),.
/*                                                                     */
/*       PRINT INTERCEPT, MULTIPLE CORRELATION COEFFICIENT, AND        */
/*       STANDARD ERROR OF ESTIMATE                                    */
/*                                                                     */
         PUT EDIT ('INTERCEPT',ANS(1),'MULTIPLE CORRELATION  ',ANS(2),
              'STD. ERROR OF ESTIMATE',ANS(3)) (SKIP(3),COLUMN(10),
              A,X(10),F(16,5),(2)(SKIP(2),COLUMN(10),A,F(13,5))),.
/*                                                                     */
/*            PRINT ANALYSIS OF VARIANCE FOR THE REGRESSION            */
/*                                                                     */
         PUT EDIT ('ANALYSIS OF VARIANCE FOR THE REGRESSION ',
              'SOURCE OF VARIATION','DEGREES','SUM OF','MEAN',
              '  F VALUE','OF FREEDOM','SQUARES','SQUARES')
              (SKIP(2),COLUMN(31),A,SKIP(2),COLUMN(15),A,X(7),A,
              X(7),A,X(10),A,X(09),A,SKIP,COLUMN(40),A,X(4),A,
              X(9),A),.
         L    =ANS(8),.
         PUT EDIT ('ATRRIBUTABLE TO REGRESSION   ',K,ANS(4),ANS(6),
              ANS(10),'DEVIATION FROM REGRESSION    ',L,ANS(7),
              ANS(9)) (SKIP,COLUMN(10),A,F(6),3 F(16,5),SKIP,
              COLUMN(10),A,F(6),2 F(16,5)),.
         L    =N-1,.
         FSUM =ANS(4)+ANS(7),.
         PUT EDIT('TOTAL',L,FSUM) (COLUMN(15),A,X(19),F(6),F(16,5)),.
         IF NRESI LE 0
         THEN GO TO S200,.
         PUT EDIT ('MULTIPLE REGRESSION.....',PR1,'SELECTION.....',I)
              (R(FM4)),.
         PUT EDIT ('TABLE OF RESIDUALS','CASE NO.','Y VALUE',
              'Y ESTIMATE','RESIDUAL') (SKIP,COLUMN(25),A,SKIP(2),
              COLUMN(10),A,X(5),A,X(5),A,X(6),A),.
         MM   =ISAVE(K+1),.
         OPEN FILE (XDATA) INPUT,.
              DO II = 1 TO N,.
              GET FILE (XDATA) EDIT ((W(J) DO J= 1 TO M))
                   ((M)F(6,0)),.
              FSUM =ANS(1),.
                   DO J = 1 TO K,.
                   L    =ISAVE(J),.
                   FSUM =FSUM+W(L)*B(J),.
                   END,.
              RESI =W(MM)-FSUM,.
              PUT EDIT (II,W(MM),FSUM,RESI) (COLUMN(10),F(5),F(15,5),
                   2 F(14,5)),.
              END,.
         CLOSE FILE (XDATA),.
   GO TO S100,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')
         (SKIP(5),COLUMN(10),A),.
S300..
   END,.                                /*END OF PROCEDURE REGR        */
```

```
DAT2..
/**********************************************************************/
/*                                                                    */
/*       TO READ FLOATING POINT DATA, ONE OBSERVATION AT A TIME.      */
/*       DATA MAY BE SAVED ON A DATA SET.                             */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,D),.
   DECLARE
        XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),
        (NCARD,NV) EXTERNAL,
        CH CHARACTER(NCARD),
        (I,M,MM) FIXED BINARY,
        D(*) FLOAT BINAPY,.
/*                                                                    */
   ON ENDFILE (SYSIN)
   GO TO EXIT,.
   GET EDIT (CH) (A(NCARD)),.
   MM   =CEIL(M/12),.
   GET STRING (CH) EDIT ((D(I) DO I= 1 TO M))
        ((MM)((12)F(6,C),X(8))),.
   IF NV= 1
   THEN PUT FILF (XDATA) EDIT ((D(I) DO I= 1 TO M)) ((M)F(6,0)),.
   FEVERT ENDFILE (SYSIN),.
   RETURN,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('ERPOR  INSUFFICIENT DATA')
        (SKIP(1),COLUMN(10),A),.
   STOP,.
   END,.                           /*END OF PROCEDURE DAT2            */
```

```
IDT1..
/**********************************************************************/
/*                                                                    */
/*       TO READ FIXED POINT DATA.                                    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,IX),.
   DECLARE
        CH CHARACTER (80),
        (IX(*),NF,N1,N2,M,I)
        FIXED BINARY,.
   NF   =40,.
   N1   =1,.
   N2   =NF,.
S10..
   IF M LE N2
   THEN N2   =M,.
   GET EDIT (CH) (A(80)),.
   GET STRING (CH) EDIT ((IX(I) DO I= N1 TO N2)) ((NF)F(2)),.
   N1   =N2+1,.
   IF N1 LE M
   THEN DO,.
        N2   =N2+NF,.
        GO TO S10,.
        END,.
   RETURN,.
   END,.                           /*END OF PROCEDURE IDT1            */
```

STEPWISE MULTIPLE REGRESSION STEP

Problem Description

Stepwise multiple 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.

1. The sample problem for stepwise multiple regression consists of 30 observations with six variables, as presented in Table 1 earlier in this Appendix.

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 the second, third, and fifth variables are used to predict the dependent variable in the second analysis.

Program

Description

The stepwise multiple regression program consists of the main routine named STEP, two special input subroutines named DAT2 and IDT2, an output subroutine named SOUT, and two routines from the Scientific Subroutine Package: CORR and STRG.

Capacity

1. Up to 99,999 observations if observations are read into the computer one at a time by the special input routine. If all data are to be stored in core before the calculation of correlation coefficients, the limitation on the number of observations depends on the size of core storage available for input data.

2. Up to 72 variables

3. Up to 99 selections (must be greater than zero)

4. (12 F(6, 0)) format for input data cards. Therefore if a problem satisfies the above conditions, the sample program need not be modified. If the input

data cards are prepared using a different format, the input format in the special input routine, DAT2, must be modified. The general rules for program modifications are described later.

Input

Control Card

One control card is required for each problem and is read by the main program, STEP. This card is prepared as follows:

Columns	Contents	For Sample Problem
1-6	Problem number (may be alphameric)	SAMPLE
7-11	Number of observations	00030
12-13	Number of variables	06
14-15	Number of selections	02
16-20	A constant value of proportion of sum of squares that will be used to limit variables entering in the regression	0.0
21	Option code for table of residuals 0 - if it is not desired 1 - if it is desired	1
22-23	Number of cards per observation	1

Leading zeros do not have to be keypunched.

Data Cards

Since input data is read into the computer one observation at a time, each row of data in table is keypunched on a separate card using the format (12 F (6, 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 next card 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 single dependent variable and a non-null set of independent variables in a stepwise multiple 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.

There must be a selection card in order for the program to continue. In the selection card each variable is specified using one of the following codes:

0 or blank - Independent variable available for selection
1 - Independent variable forced in regression
2 - Variable to be deleted
3 - Dependent variable

		For Sample Problem	
Columns	Contents	Selection 1	Selection 2
1	First variable	0	2
2	Second variable	0	0
3	Third variable	0	0
4	Fourth variable	0	2
5	Fifth variable	0	0
6	Sixth variable	3	3
	.		
	.		
	.		
72	72nd variable		

Leading zeros do not have to be keypunched. If more than 72 selections are made, continue selection specification codes beginning in column 1 of a second card.

Deck Setup

Deck setup is shown in Figure 17.

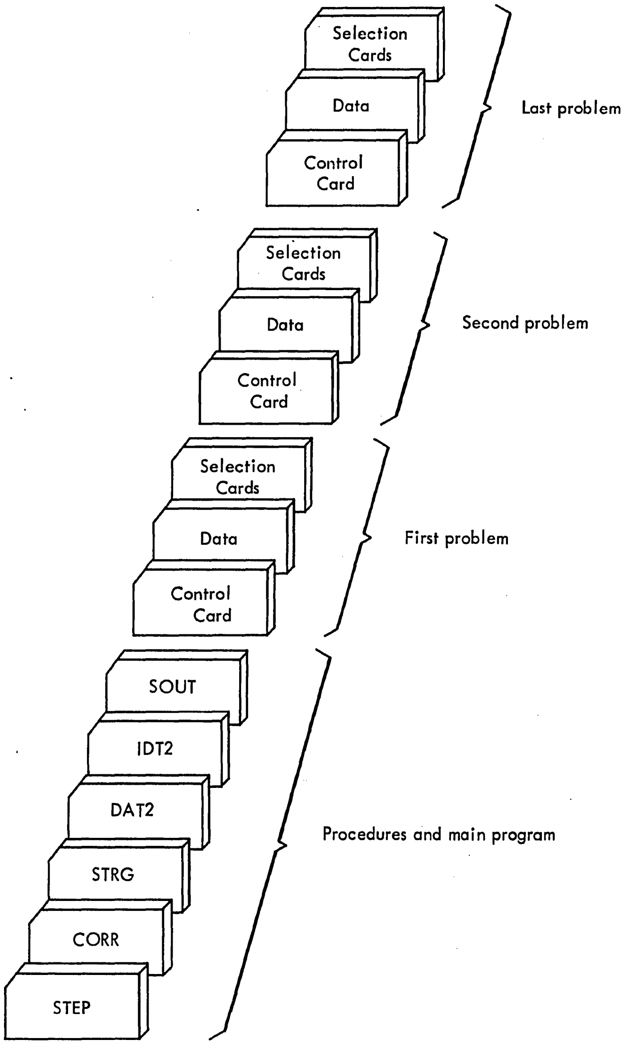
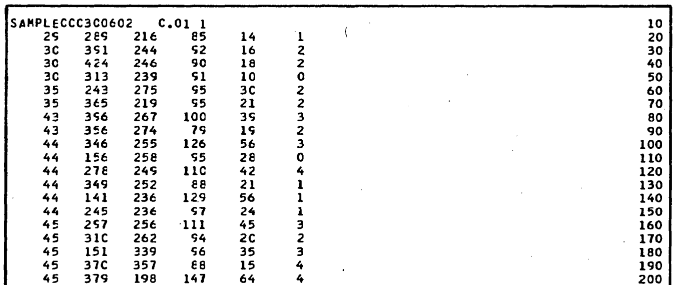


Figure 17.

Sample

The listing of the input cards for the sample problem is shown in Figure 18.



```
SAMPLE0003C0602    0.01 1                                             10
    29    289    216     85     14      1                             20
    30    351    244     92     16      2                             30
    30    424    246     90     18      2                             40
    30    313    239     91     10      0                             50
    35    243    275     95     30      2                             60
    35    365    219     95     21      2                             70
    43    396    267    100     39      3                             80
    43    356    274     79     19      2                             90
    44    346    255    126     56      3                            100
    44    156    258     95     28      0                            110
    44    278    249    110     42      4                            120
    44    349    252     88     21      1                            130
    44    141    236    129     56      1                            140
    44    245    236     97     24      1                            150
    45    297    256    111     45      3                            160
    45    310    262     94     20      2                            170
    45    151    339     96     35      3                            180
    45    370    357     88     15      4                            190
    45    379    198    147     64      4                            200
    45    463    206    105     31      3                            210
    45    316    245    132     60      4                            220
    45    280    225    108     36      4                            230
    44    395    215    101     27      1                            240
    49    139    220    136     59      0                            250
    49    245    205    113     37      4                            260
    49    373    215     88     29      1                            270
    51    224    215    118     54      3                            280
    51    677    210    116     33      4                            290
    51    424    210    140     59      4                            300
    51    150    210    105     30      0                            310
000003                                                               320
200203                                                               330
```

Figure 18.

Output

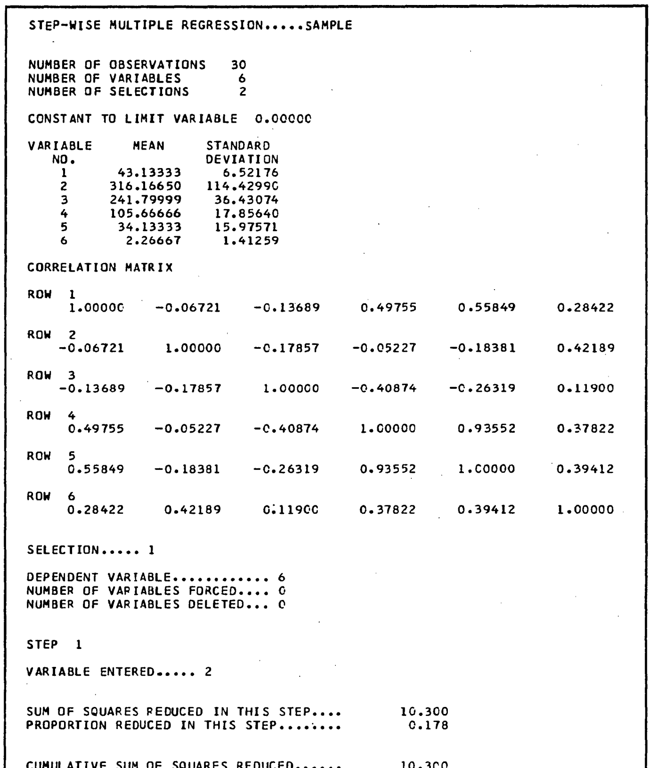
Description

The output of the sample program for stepwise multiple regression includes:

1. Means
2. Standard deviations
3. Correlation coefficients between independent variables and dependent variables
4. Sum of squares reduced in the step
5. Proportion reduced in the step
6. Multiple correlation coefficient
7. F value for analysis of variance
8. Standard error of estimate
9. Computed T value
10. Beta coefficients
11. Table of residuals (optional)

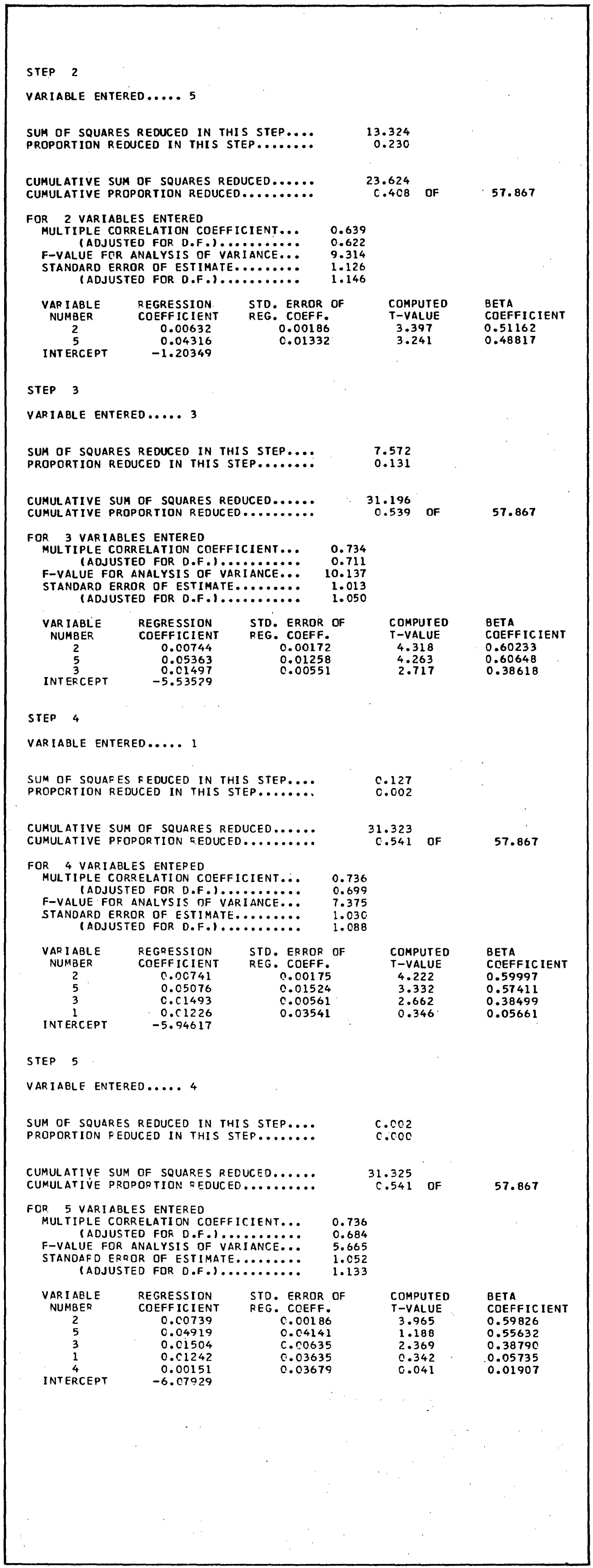
Sample

The output listing for the sample problem is shown in Figure 19.



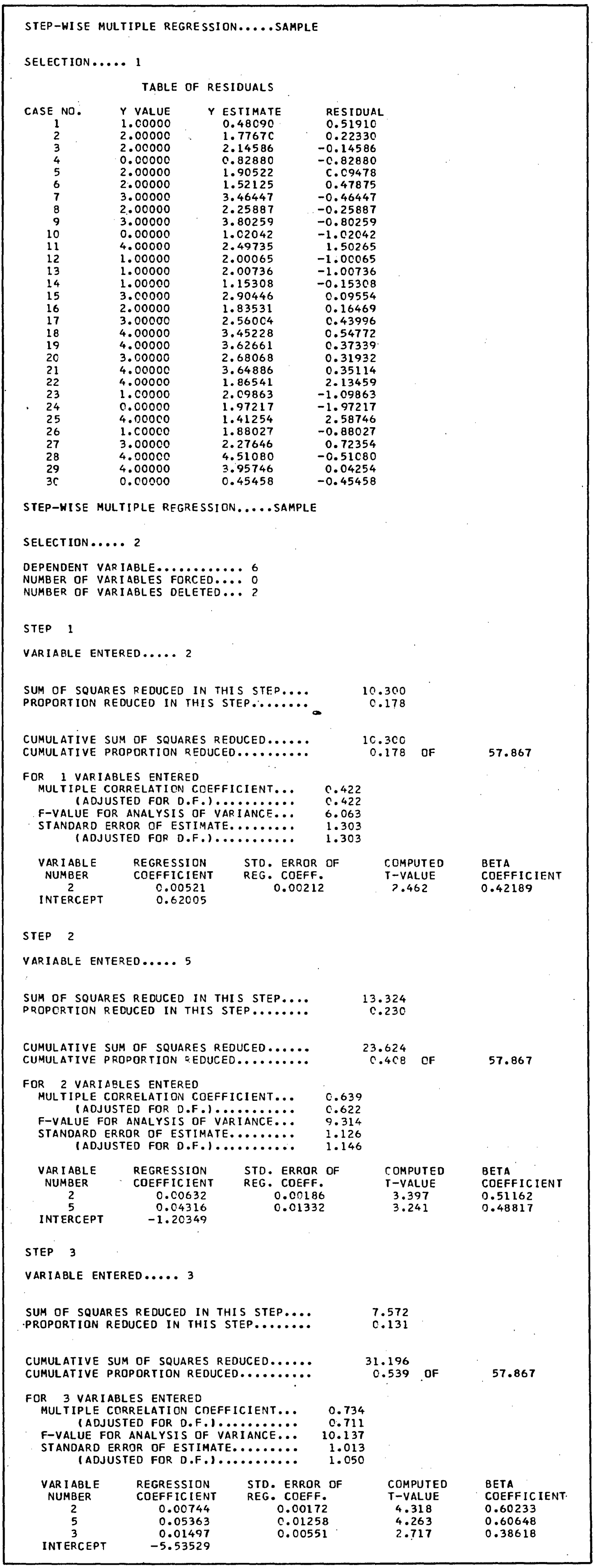
```
STEP-WISE MULTIPLE REGRESSION.....SAMPLE


NUMBER OF OBSERVATIONS   30
NUMBER OF VARIABLES       6
NUMBER OF SELECTIONS      2

CONSTANT TO LIMIT VARIABLE  0.00000

VARIABLE      MEAN      STANDARD
   NO.                  DEVIATION
    1       43.13333     6.52176
    2      316.16650   114.42990
    3      241.79999    36.43074
    4      105.66666    17.85640
    5       34.13333    15.97571
    6        2.26667     1.41259

CORRELATION MATRIX

ROW  1
     1.00000    -0.06721    -0.13689     0.49755     0.55849     0.28422

ROW  2
    -0.06721     1.00000    -0.17857    -0.05227    -0.18381     0.42189

ROW  3
    -0.13689    -0.17857     1.00000    -0.40874    -0.26319     0.11900

ROW  4
     0.49755    -0.05227    -0.40874     1.00000     0.93552     0.37822

ROW  5
     0.55849    -0.18381    -0.26319     0.93552     1.00000     0.39412

ROW  6
     0.28422     0.42189     0.11900     0.37822     0.39412     1.00000


SELECTION..... 1

DEPENDENT VARIABLE............ 6
NUMBER OF VARIABLES FORCED.... 0
NUMBER OF VARIABLES DELETED... 0


STEP  1

VARIABLE ENTERED..... 2


SUM OF SQUARES REDUCED IN THIS STEP....        10.300
PROPORTION REDUCED IN THIS STEP........         0.178


CUMULATIVE SUM OF SQUARES REDUCED......        10.300
CUMULATIVE PROPORTION REDUCED..........         0.178  OF        57.867

FOR  1 VARIABLES ENTERED
  MULTIPLE CORRELATION COEFFICIENT...    0.422
       (ADJUSTED FOR D.F.)...........    0.422
  F-VALUE FOR ANALYSIS OF VARIANCE...    6.063
  STANDARD ERROR OF ESTIMATE.........    1.303
       (ADJUSTED FOR D.F.)...........    1.303

  VARIABLE      REGRESSION     STD. ERROR OF      COMPUTED     BETA
   NUMBER       COEFFICIENT    REG. COEFF.         T-VALUE     COEFFICIENT
     2            0.00521         0.00212           2.462      0.42189
  INTERCEPT       0.62005
```

Figure 19.



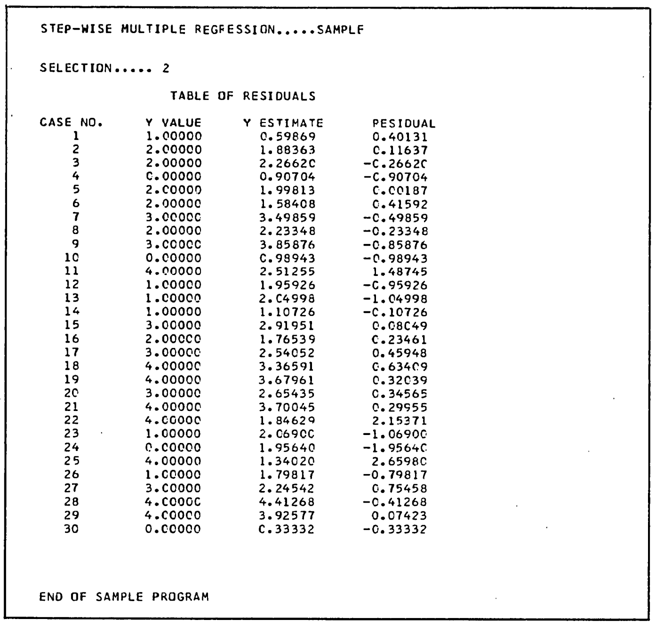
```
STEP  2

VARIABLE ENTERED..... 5


SUM OF SQUARES REDUCED IN THIS STEP....         13.324
PROPORTION REDUCED IN THIS STEP........          0.230


CUMULATIVE SUM OF SQUARES REDUCED......         23.624
CUMULATIVE PROPORTION REDUCED..........          C.4C8   OF          57.867

FOR  2 VARIABLES ENTERED
  MULTIPLE CORRELATION COEFFICIENT...      0.639
       (ADJUSTED FOR D.F.)...........      0.622
  F-VALUE FCR ANALYSIS OF VARIANCE...      9.314
  STANDARD ERROR OF ESTIMATE.........      1.126
       (ADJUSTED FOR D.F.)...........      1.146

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             0.00632         0.00186           3.397          0.51162
      5             0.04316         C.01332           3.241          0.48817
  INTERCEPT        -1.20349

STEP  3

VARIABLE ENTERED..... 3


SUM OF SQUARES REDUCED IN THIS STEP....          7.572
PROPORTION REDUCED IN THIS STEP........          0.131


CUMULATIVE SUM OF SQUARES REDUCED......         31.196
CUMULATIVE PROPORTION REDUCED..........          0.539   OF          57.867

FOR  3 VARIABLES ENTERED
  MULTIPLE CORRELATION COEFFICIENT...      0.734
       (ADJUSTED FOR D.F.)...........      0.711
  F-VALUE FOR ANALYSIS OF VARIANCE...     10.137
  STANDARD ERROR OF ESTIMATE.........      1.013
       (ADJUSTED FOR D.F.)...........      1.050

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             0.C0744         0.00172           4.318          0.60233
      5             0.05363         0.01258           4.263          0.60648
      3             0.C1497         C.00551           2.717          0.38618
  INTERCEPT        -5.53529

STEP  4

VARIABLE ENTERED..... 1


SUM OF SQUAFES FEDUCED IN THIS STEP....          C.127
PROPCRTION REDUCED IN THIS STEP........          C.002


CUMULATIVE SUM OF SQUARES REDUCED......         31.323
CUMULATIVE PFOPORTION REDUCED..........          C.541   OF          57.867

FOR  4 VARIABLES ENTEPED
  MULTIPLE CORRELATION COEFFICIENT...      0.736
       (ADJUSTED FOR D.F.)...........      0.699
  F-VALUE FOR ANALYSIS OF VARIANCE...      7.375
  STANDARD ERROR OF ESTIMATE.........      1.03C
       (ADJUSTED FOR D.F.)...........      1.088

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             C.00741         0.00175           4.222          0.59997
      5             0.05076         0.01524           3.332          0.57411
      3             C.C1493         C.00561           2.662          0.38499
      1             0.C1226         0.03541           0.346          0.05661
  INTERCEPT        -5.94617

STEP  5

VARIABLE ENTERED..... 4


SUM OF SQUARES REDUCED IN THIS STEP....          C.CO2
PROPORTION FEDUCED IN THIS STEP........          C.COC


CUMULATIVE SUM OF SQUARES REDUCED......         31.325
CUMULATIVE PROPORTION REDUCED..........          C.541   OF          57.867

FOR  5 VARIABLES ENTERED
  MULTIPLE CORRELATION COEFFICIENT...      0.736
       (ADJUSTED FOR D.F.)...........      0.684
  F-VALUE FOR ANALYSIS OF VARIANCE...      5.665
  STANDAFD ERROR OF ESTIMATE.........      1.052
       (ADJUSTED FOR D.F.)...........      1.133

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             0.C0739         C.00186           3.965          0.59826
      5             C.04919         0.C4141           1.188          0.55632
      3             0.01504         C.C0635           2.369          0.3879C
      1             0.C1242         C.03635           C.342          0.05735
      4             0.00151         0.03679           C.041          0.01907
  INTERCEPT        -6.07929
```

Figure 19. (Continued)



```
STEP-WISE MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 1

                 TABLE OF RESIDUALS

CASE NO.       Y VALUE       Y ESTIMATE        RESIDUAL
    1          1.C0000        0.48C90          0.5191C
    2          2.0000C        1.7767C          0.2233C
    3          2.0C000        2.14586         -0.14586
    4          0.0000C        C.82880         -C.82880
    5          2.00000        1.90522          C.C9478
    6          2.00000        1.52125          0.47875
    7          3.0000C        3.46447         -0.46447
    8          2.00000        2.25887         -0.25887
    9          3.00000        3.80259         -0.80259
   10          0.00000        1.C2042         -1.C2042
   11          4.00000        2.49735          1.50265
   12          1.00000        2.00065         -1.0C065
   13          1.00000        2.00736         -1.00736
   14          1.00000        1.15308         -0.15308
   15          3.C0000        2.90446          0.09554
   16          2.00000        1.83531          0.16469
   17          3.00000        2.560C4          C.43996
   18          4.00000        3.45228          0.54772
   19          4.00000        3.62661          0.37339
   2C          3.00000        2.68068          0.31932
   21          4.0000C        3.64886          0.35114
   22          4.00000        1.86541          2.13459
   23          1.C0000        2.C9863         -1.09863
   24          0.00000        1.97217         -1.97217
   25          4.0000C        1.41254          2.58746
   26          1.C000C        1.88027         -0.88027
   27          3.0000C        2.27646          0.72354
   28          4.0000C        4.51080         -0.51C80
   29          4.00000        3.95746          0.04254
   3C          0.C0000        0.45458         -0.45458

STEP-WISE MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 2

DEPENDENT VARIABLE............ 6
NUMBER OF VARIABLES FORCED.... 0
NUMBER OF VARIABLES DELETED... 2


STEP  1

VARIABLE ENTERED..... 2


SUM OF SQUARES REDUCED IN THIS STEP....         10.300
PROPORTION REDUCED IN THIS STEP........          C.178


CUMULATIVE SUM OF SQUARES REDUCED......         1C.3CC
CUMULATIVE PROPORTION REDUCED..........          0.178   OF          57.867

FOR  1 VARIABLES ENTERED
  MULTIPLE CORRELATICN COEFFICIENT...      C.422
       (ADJUSTED FOR D.F.)...........      C.422
  F-VALUE FOR ANALYSIS OF VARIANCE...      6.063
  STANDARD ERROR OF ESTIMATE.........      1.303
       (ADJUSTED FOR D.F.)...........      1.303

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             C.00521         0.00212           ?.462          0.42189
  INTERCEPT         0.62005

STEP  2

VARIABLE ENTERED..... 5


SUM OF SQUARES REDUCED IN THIS STEP....         13.324
PROPCRTION REDUCED IN THIS STEP........          C.23C


CUMULATIVE SUM OF SQUARES REDUCED......         23.624
CUMULATIVE PROPORTION REDUCED..........          C.4C8   CF          57.867

FOR  2 VARIABLES ENTERED
  MULTIPLE CORRELATION COEFFICIENT...      C.639
       (ADJUSTED FOR D.F.)...........      C.622
  F-VALUE FOR ANALYSIS OF VARIANCE...      9.314
  STANDARD ERROR OF ESTIMATE.........      1.126
       (ADJUSTED FOR D.F.)...........      1.146

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             0.C0632         0.00186           3.397          0.51162
      5             0.04316         0.01332           3.241          0.48817
  INTERCEPT        -1.20349

STEP  3

VARIABLE ENTERED..... 3


SUM OF SQUARES REDUCED IN THIS STEP....          7.572
PROPORTION REDUCED IN THIS STEP........          C.131


CUMULATIVE SUM OF SQUARES REDUCED......         31.196
CUMULATIVE PROPORTION REDUCED..........          0.539   OF          57.867

FOR  3 VARIABLES ENTERED
  MULTIPLE CORRELATION COEFFICIENT...      0.734
       (ADJUSTED FOR D.F.)...........      C.711
  F-VALUE FOR ANALYSIS OF VARIANCE...     10.137
  STANDARD ERROR OF ESTIMATE.........      1.013
       (ADJUSTED FOR D.F.)...........      1.050

  VARIABLE       REGRESSION     STD. ERROR OF        COMPUTED        BETA
   NUMBER        COEFFICIENT    REG. COEFF.          T-VALUE         COEFFICIENT
      2             0.00744         0.00172           4.318          0.60233
      5             0.05363         C.01258           4.263          0.60648
      3             0.01497         0.00551           2.717          0.38618
  INTERCEPT        -5.53529
```

Figure 19. (Continued)



```
STEP-WISE MULTIPLE REGFESSION.....SAMPLF

SELECTION..... 2

                    TABLE OF RESIDUALS

CASE NO.        Y VALUE       Y ESTIMATE      PESIDUAL
     1          1.00000         0.59869        0.40131
     2          2.00000         1.88363        C.11637
     3          2.00000         2.2662C       -C.2662C
     4          C.00000         0.90704       -C.90704
     5          2.C0000         1.99813        C.C0187
     6          2.00000         1.58408        C.41592
     7          3.0000C         3.49859       -C.49859
     8          2.00000         2.23348       -0.23348
     9          3.C000C         3.85876       -0.85876
    1C          0.00000         C.98943       -0.98943
    11          4.00000         2.51255        1.48745
    12          1.00000         1.95926       -C.95926
    13          1.C0000         2.C4998       -1.04998
    14          1.00000         1.10726       -C.10726
    15          3.00000         2.91951        0.08C49
    16          2.0000C         1.76539        C.23461
    17          3.0000C         2.54052        0.45948
    18          4.0000C         3.36591        C.63409
    19          4.00000         3.67961        0.32039
    2C          3.00000         2.65435        0.34565
    21          4.00000         3.70045        0.29955
    22          4.C000C         1.84629        2.15371
    23          1.00000         2.0690C       -1.0690C
    24          0.C0000         1.95640       -1.9564C
    25          4.00000         1.34020        2.6598C
    26          1.C0000         1.79817       -0.79817
    27          3.C0000         2.24542        0.75458
    28          4.C000C         4.41268       -C.41268
    29          4.C000C         3.92577        0.07423
    30          0.C0000         C.33332       -0.33332



END OF SAMPLE PROGRAM
```

Figure 19. (Continued)

Program Modifications

Input data in a different format can be handled by providing a special format statement. The special input routine, DAT2 is normally written by the user to handle different formats for different problems. The user may modify this routine to perform testing of input data, transformation of data and so on. When doing so, attention should be paid to the format statement in DAT2 (DAT2 230), which writes on the intermediate data set. The format in this statement must be the same as the format in statement STEP 1390.

Operating Instructions

The sample program for stepwise multiple regression is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output. A scratch tape (data set XDATA) is used as intermediate storage.

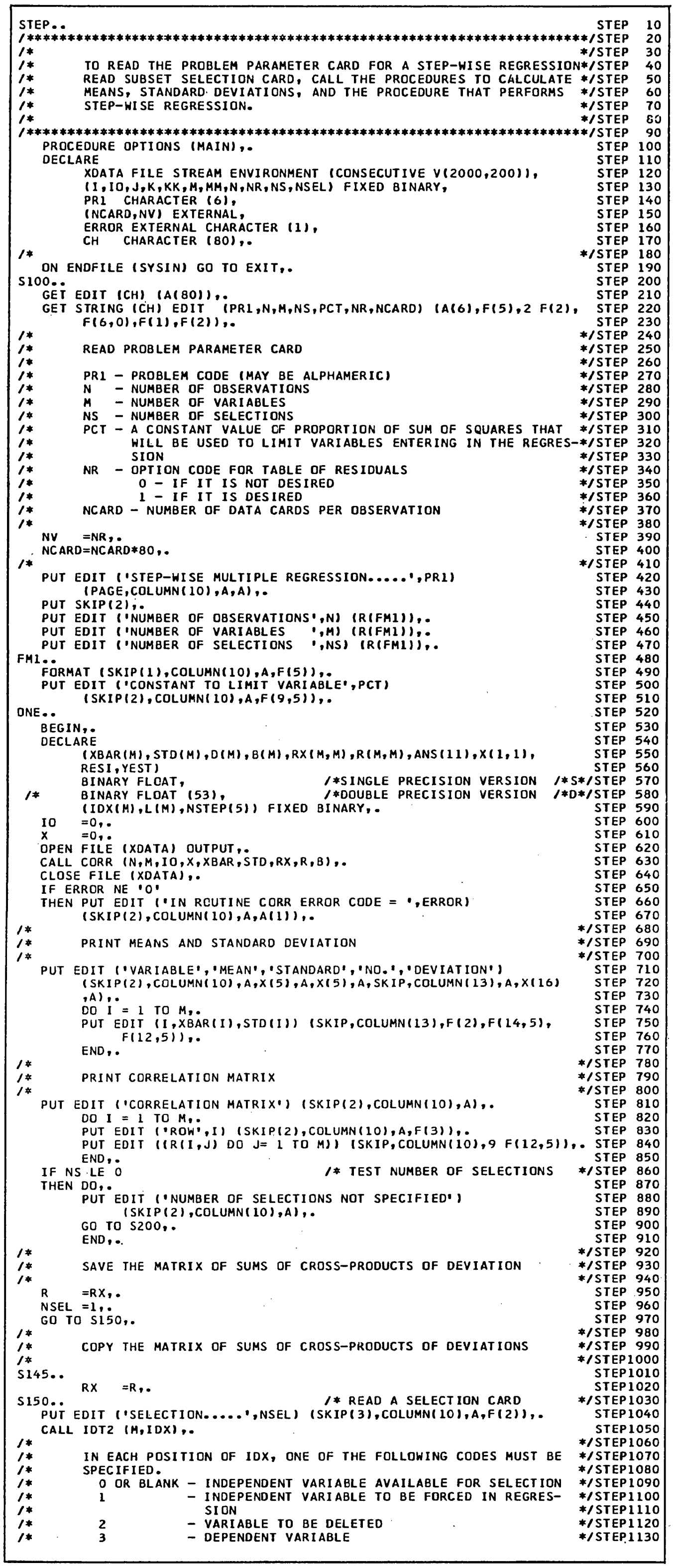
Error Messages

The following error condition will result in a message:

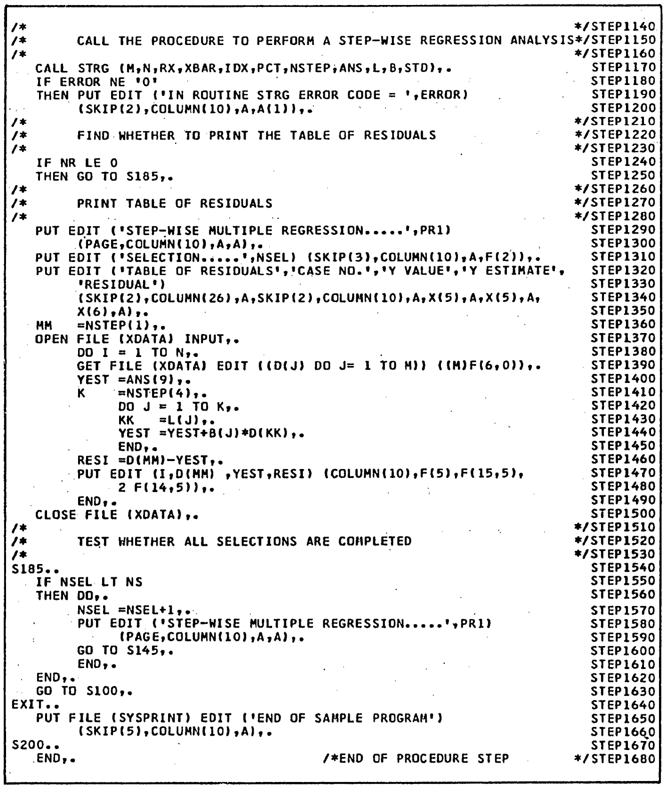
1. The number of selections not specified on the control card: NUMBER OF SELECTIONS NOT SPECIFIED. JOB TERMINATED.

Timing

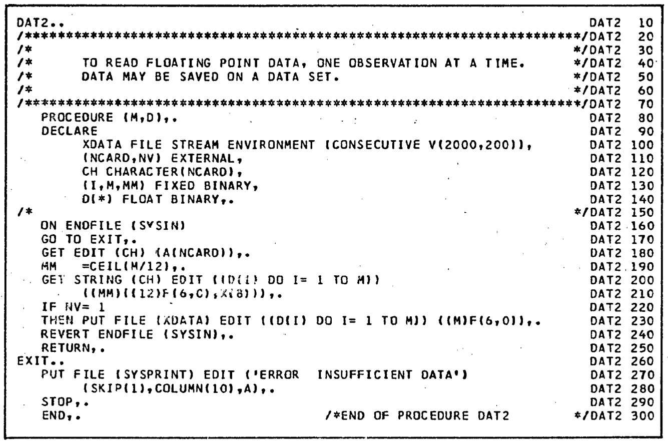
The execution of this sample program on a System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 41 seconds.



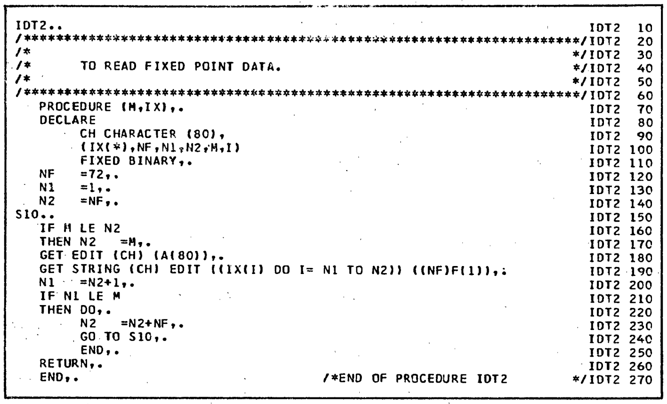
```
STEP..                                                                  STEP  10
/***********************************************************************/STEP  20
/*                                                                     */STEP  30
/*       TO READ THE PROBLEM PARAMETER CARD FOR A STEP-WISE REGRESSION*/STEP  40
/*       READ SUBSET SELECTION CARD, CALL THE PROCEDURES TO CALCULATE */STEP  50
/*       MEANS, STANDARD DEVIATIONS, AND THE PROCEDURE THAT PERFORMS  */STEP  60
/*       STEP-WISE REGRESSION.                                        */STEP  70
/*                                                                    */STEP  80
/***********************************************************************/STEP  90
   PROCEDURE OPTIONS (MAIN),.                                           STEP 100
   DECLARE                                                              STEP 110
        XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),        STEP 120
        (I,IO,J,K,KK,M,MM,N,NR,NS,NSEL) FIXED BINARY,                   STEP 130
        PR1  CHARACTER (6),                                             STEP 140
        (NCARD,NV) EXTERNAL,                                            STEP 150
        ERROR EXTERNAL CHARACTER (1),                                   STEP 160
        CH   CHARACTER (80),.                                           STEP 170
/*                                                                    */STEP 180
   ON ENDFILE (SYSIN) GO TO EXIT,.                                      STEP 190
S100..                                                                  STEP 200
   GET EDIT (CH) (A(80)),.                                              STEP 210
   GET STRING (CH) EDIT  (PR1,N,M,NS,PCT,NR,NCARD) (A(6),F(5),2 F(2),   STEP 220
        F(6,0),F(1),F(2)),.                                             STEP 230
/*                                                                    */STEP 240
/*      READ PROBLEM PARAMETER CARD                                   */STEP 250
/*                                                                    */STEP 260
/*      PR1 - PROBLEM CODE (MAY BE ALPHAMERIC)                        */STEP 270
/*      N   - NUMBER OF OBSERVATIONS                                  */STEP 280
/*      M   - NUMBER OF VARIABLES                                     */STEP 290
/*      NS  - NUMBER OF SELECTIONS                                    */STEP 300
/*      PCT - A CONSTANT VALUE OF PROPORTION OF SUM OF SQUARES THAT   */STEP 310
/*            WILL BE USED TO LIMIT VARIABLES ENTERING IN THE REGRES-*/STEP 320
/*            SION                                                    */STEP 330
/*      NR  - OPTION CODE FOR TABLE OF RESIDUALS                      */STEP 340
/*             0 - IF IT IS NOT DESIRED                               */STEP 350
/*             1 - IF IT IS DESIRED                                   */STEP 360
/*      NCARD - NUMBER OF DATA CARDS PER OBSERVATION                  */STEP 370
/*                                                                    */STEP 380
   NV   =NR,.                                                           STEP 390
   NCARD=NCARD*80,.                                                     STEP 400
/*                                                                    */STEP 410
   PUT EDIT ('STEP-WISE MULTIPLE REGRESSION.....',PR1)                  STEP 420
        (PAGE,COLUMN(10),A,A),.                                         STEP 430
   PUT SKIP(2),.                                                        STEP 440
   PUT EDIT ('NUMBER OF OBSERVATIONS',N) (R(FM1)),.                     STEP 450
   PUT EDIT ('NUMBER OF VARIABLES   ',M) (R(FM1)),.                     STEP 460
   PUT EDIT ('NUMBER OF SELECTIONS  ',NS) (R(FM1)),.                    STEP 470
FM1..                                                                   STEP 480
   FORMAT (SKIP(1),COLUMN(10),A,F(5)),.                                 STEP 490
   PUT EDIT ('CONSTANT TO LIMIT VARIABLE',PCT)                          STEP 500
        (SKIP(2),COLUMN(10),A,F(9,5)),.                                 STEP 510
ONE..                                                                   STEP 520
   BEGIN,.                                                              STEP 530
   DECLARE                                                              STEP 540
        (XBAR(M),STD(M),D(M),B(M),RX(M,M),R(M,M),ANS(11),X(1,1),        STEP 550
        RESI,YEST)                                                      STEP 560
        BINARY FLOAT,                    /*SINGLE PRECISION VERSION  /*S*/STEP 570
 /*     BINARY FLOAT (53),               /*DOUBLE PRECISION VERSION  /*D*/STEP 580
        (IDX(M),L(M),NSTEP(5)) FIXED BINARY,.                           STEP 590
   IO   =0,.                                                            STEP 600
   X    =0,.                                                            STEP 610
   OPEN FILE (XDATA) OUTPUT,.                                           STEP 620
   CALL CORR (N,M,IO,X,XBAR,STD,RX,R,B),.                               STEP 630
   CLOSE FILE (XDATA),.                                                 STEP 640
   IF ERROR NE '0'                                                      STEP 650
   THEN PUT EDIT ('IN ROUTINE CORR ERROR CODE = ',ERROR)                STEP 660
        (SKIP(2),COLUMN(10),A,A(1)),.                                   STEP 670
/*                                                                    */STEP 680
/*      PRINT MEANS AND STANDARD DEVIATION                            */STEP 690
/*                                                                    */STEP 700
   PUT EDIT ('VARIABLE','MEAN','STANDARD','NO.','DEVIATION')            STEP 710
        (SKIP(2),COLUMN(10),A,X(5),A,X(5),A,SKIP,COLUMN(13),A,X(16)     STEP 720
        ,A),.                                                           STEP 730
        DO I = 1 TO M,.                                                 STEP 740
        PUT EDIT (I,XBAR(I),STD(I)) (SKIP,COLUMN(13),F(2),F(14,5),      STEP 750
             F(12,5)),.                                                 STEP 760
        END,.                                                           STEP 770
/*                                                                    */STEP 780
/*      PRINT CORRELATION MATRIX                                      */STEP 790
/*                                                                    */STEP 800
   PUT EDIT ('CORRELATION MATRIX') (SKIP(2),COLUMN(10),A),.             STEP 810
        DO I = 1 TO M,.                                                 STEP 820
        PUT EDIT ('ROW',I) (SKIP(2),COLUMN(10),A,F(3)),.                STEP 830
        PUT EDIT ((R(I,J) DO J= 1 TO M)) (SKIP,COLUMN(10),9 F(12,5)),. STEP 840
        END,.                                                           STEP 850
   IF NS LE 0                         /* TEST NUMBER OF SELECTIONS    */STEP 860
   THEN DO,.                                                            STEP 870
        PUT EDIT ('NUMBER OF SELECTIONS NOT SPECIFIED')                 STEP 880
             (SKIP(2),COLUMN(10),A),.                                   STEP 890
        GO TO S200,.                                                    STEP 900
        END,.                                                           STEP 910
/*                                                                    */STEP 920
/*      SAVE THE MATRIX OF SUMS OF CROSS-PRODUCTS OF DEVIATION        */STEP 930
/*                                                                    */STEP 940
   R    =RX,.                                                           STEP 950
   NSEL =1,.                                                            STEP 960
   GO TO S150,.                                                         STEP 970
/*                                                                    */STEP 980
/*      COPY THE MATRIX OF SUMS OF CROSS-PRODUCTS OF DEVIATIONS       */STEP 990
/*                                                                    */STEP1000
S145..                                                                  STEP1010
        RX   =R,.                                                       STEP1020
S150..                                /* READ A SELECTION CARD        */STEP1030
   PUT EDIT ('SELECTION.....',NSEL) (SKIP(3),COLUMN(10),A,F(2)),.       STEP1040
   CALL IDT2 (M,IDX),.                                                  STEP1050
/*                                                                    */STEP1060
/*      IN EACH POSITION OF IDX, ONE OF THE FOLLOWING CODES MUST BE   */STEP1070
/*      SPECIFIED.                                                    */STEP1080
/*        0 OR BLANK - INDEPENDENT VARIABLE AVAILABLE FOR SELECTION   */STEP1090
/*        1          - INDEPENDENT VARIABLE TO BE FORCED IN REGRES-   */STEP1100
/*                     SION                                           */STEP1110
/*        2          - VARIABLE TO BE DELETED                         */STEP1120
/*        3          - DEPENDENT VARIABLE                             */STEP1130
```



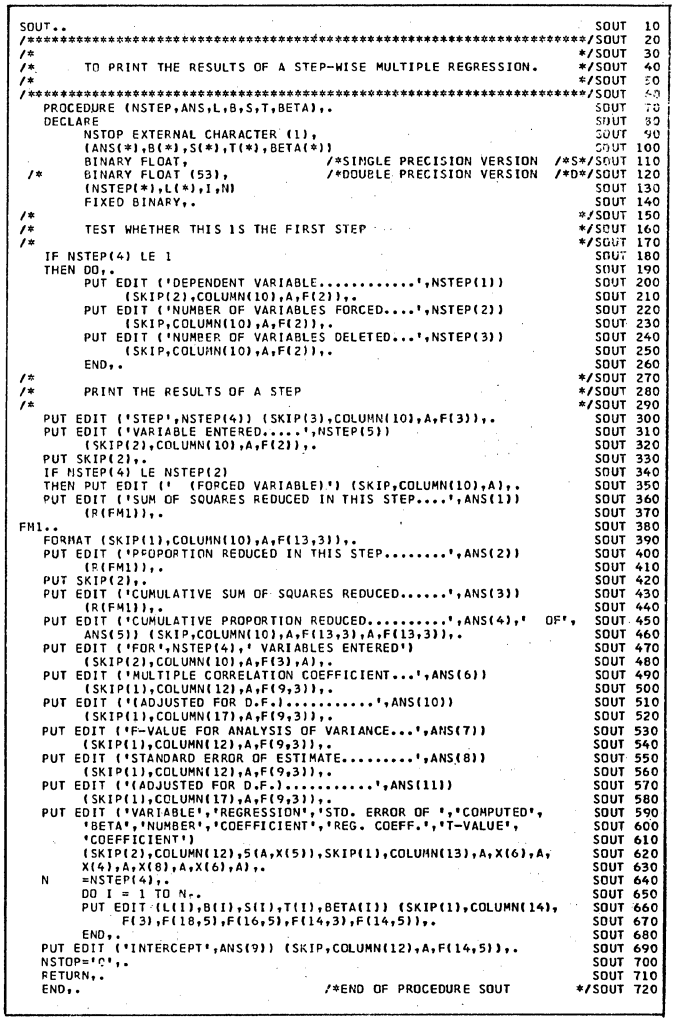
```
/*                                                                      */
/*      CALL THE PROCEDURE TO PERFORM A STEP-WISE REGRESSION ANALYSIS*/
/*                                                                      */
   CALL STRG (M,N,RX,XBAR,IDX,PCT,NSTEP,ANS,L,B,STD),.
   IF ERROR NE '0'
   THEN PUT EDIT ('IN ROUTINE STRG ERROR CODE = ',ERROR)
         (SKIP(2),COLUMN(10),A,A(1)),.
/*                                                                      */
/*      FIND WHETHER TO PRINT THE TABLE OF RESIDUALS                    */
/*                                                                      */
   IF NR LE 0
   THEN GO TO S185,.
/*                                                                      */
/*      PRINT TABLE OF RESIDUALS                                        */
/*                                                                      */
   PUT EDIT ('STEP-WISE MULTIPLE REGRESSION.....',PR1)
         (PAGE,COLUMN(10),A,A),.
   PUT EDIT ('SELECTION.....',NSEL) (SKIP(3),COLUMN(10),A,F(2)),.
   PUT EDIT ('TABLE OF RESIDUALS','CASE NO.','Y VALUE','Y ESTIMATE',
         'RESIDUAL')
         (SKIP(2),COLUMN(26),A,SKIP(2),COLUMN(10),A,X(5),A,X(5),A,
         X(6),A),.
   MM    =NSTEP(1),.
   OPEN FILE (XDATA) INPUT,.
         DO I = 1 TO N,.
         GET FILE (XDATA) EDIT ((D(J) DO J= 1 TO M)) ((M)F(6,0)),.
         YEST =ANS(9),.
         K    =NSTEP(4),.
               DO J = 1 TO K,.
               KK    =L(J),.
               YEST =YEST+B(J)*D(KK),.
               END,.
         RESI =D(MM)-YEST,.
         PUT EDIT (I,D(MM) ,YEST,RESI) (COLUMN(10),F(5),F(15,5),
               2 F(14,5)),.
         END,.
   CLOSE FILE (XDATA),.
/*                                                                      */
/*      TEST WHETHER ALL SELECTIONS ARE COMPLETED                       */
/*                                                                      */
S185..
   IF NSEL LT NS
   THEN DO,.
         NSEL =NSEL+1,.
         PUT EDIT ('STEP-WISE MULTIPLE REGRESSION.....',PR1)
               (PAGE,COLUMN(10),A,A),.
         GO TO S145,.
         END,.
   END,.
   GO TO S100,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')
         (SKIP(5),COLUMN(10),A),.
S200..
   END,.                                      /*END OF PROCEDURE STEP  */
```



```
DAT2..
/***********************************************************************/
/*                                                                      */
/*      TO READ FLOATING POINT DATA, ONE OBSERVATION AT A TIME.        */
/*      DATA MAY BE SAVED ON A DATA SET.                                */
/*                                                                      */
/***********************************************************************/
   PROCEDURE (M,D),.
   DECLARE
         XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),
         (NCARD,NV) EXTERNAL,
         CH CHARACTER(NCARD),
         (I,M,MM) FIXED BINARY,
         D(*) FLOAT BINARY,.
/*                                                                      */
   ON ENDFILE (SYSIN)
   GO TO EXIT,.
   GET EDIT (CH) (A(NCARD)),.
   MM    =CEIL(M/12),.
   GET STRING (CH) EDIT ((D(I) DO I= 1 TO M))
         ((MM)((12)F(6,0),X(8))),.
   IF NV= 1
   THEN PUT FILE (XDATA) EDIT ((D(I) DO I= 1 TO M)) ((M)F(6,0)),.
   REVERT ENDFILE (SYSIN),.
   RETURN,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('ERROR  INSUFFICIENT DATA')
         (SKIP(1),COLUMN(10),A),.
   STOP,.
   END,.                                      /*END OF PROCEDURE DAT2  */
```



```
IDT2..
/***********************************************************************/
/*                                                                      */
/*      TO READ FIXED POINT DATA.                                       */
/*                                                                      */
/***********************************************************************/
   PROCEDURE (M,IX),.
   DECLARE
         CH CHARACTER (80),
         (IX(*),NF,N1,N2,M,I)
         FIXED BINARY,.
   NF    =72,.
   N1    =1,.
   N2    =NF,.
S10..
   IF M LE N2
   THEN N2    =M,.
   GET EDIT (CH) (A(80)),.
   GET STRING (CH) EDIT ((IX(I) DO I= N1 TO N2)) ((NF)F(1)),:
   N1    =N2+1,.
   IF N1 LE M
   THEN DO,.
         N2    =N2+NF,.
         GO TO S10,.
         END,.
   RETURN,.
   END,.                                      /*END OF PROCEDURE IDT2  */
```



```
SOUT..
/***********************************************************************/
/*                                                                      */
/*      TO PRINT THE RESULTS OF A STEP-WISE MULTIPLE REGRESSION.       */
/*                                                                      */
/***********************************************************************/
   PROCEDURE (NSTEP,ANS,L,B,S,T,BETA),.
   DECLARE
         NSTOP EXTERNAL CHARACTER (1),
         (ANS(*),B(*),S(*),T(*),BETA(*))
         BINARY FLOAT,                /*SINGLE PRECISION VERSION  /*S*/
 /*      BINARY FLOAT (53),           /*DOUBLE PRECISION VERSION  /*D*/
         (NSTEP(*),L(*),I,N)
         FIXED BINARY,.
/*                                                                      */
/*      TEST WHETHER THIS IS THE FIRST STEP                             */
/*                                                                      */
   IF NSTEP(4) LE 1
   THEN DO,.
         PUT EDIT ('DEPENDENT VARIABLE............',NSTEP(1))
               (SKIP(2),COLUMN(10),A,F(2)),.
         PUT EDIT ('NUMBER OF VARIABLES FORCED....',NSTEP(2))
               (SKIP,COLUMN(10),A,F(2)),.
         PUT EDIT ('NUMBER OF VARIABLES DELETED...',NSTEP(3))
               (SKIP,COLUMN(10),A,F(2)),.
         END,.
/*                                                                      */
/*      PRINT THE RESULTS OF A STEP                                     */
/*                                                                      */
   PUT EDIT ('STEP',NSTEP(4)) (SKIP(3),COLUMN(10),A,F(3)),.
   PUT EDIT ('VARIABLE ENTERED.....',NSTEP(5))
         (SKIP(2),COLUMN(10),A,F(2)),.
   PUT SKIP(2),.
   IF NSTEP(4) LE NSTEP(2)
   THEN PUT EDIT ('  (FORCED VARIABLE)') (SKIP,COLUMN(10),A),.
   PUT EDIT ('SUM OF SQUARES REDUCED IN THIS STEP....',ANS(1))
         (R(FM1)),.
FM1..
   FORMAT (SKIP(1),COLUMN(10),A,F(13,3)),.
   PUT EDIT ('PROPORTION REDUCED IN THIS STEP........',ANS(2))
         (R(FM1)),.
   PUT SKIP(2),.
   PUT EDIT ('CUMULATIVE SUM OF SQUARES REDUCED......',ANS(3))
         (R(FM1)),.
   PUT EDIT ('CUMULATIVE PROPORTION REDUCED..........',ANS(4),'  OF',
         ANS(5)) (SKIP,COLUMN(10),A,F(13,3),A,F(13,3)),.
   PUT EDIT ('FOR',NSTEP(4),' VARIABLES ENTERED')
         (SKIP(2),COLUMN(10),A,F(3),A),.
   PUT EDIT ('MULTIPLE CORRELATION COEFFICIENT...',ANS(6))
         (SKIP(1),COLUMN(12),A,F(9,3)),.
   PUT EDIT ('(ADJUSTED FOR D.F.)...........',ANS(10))
         (SKIP(1),COLUMN(17),A,F(9,3)),.
   PUT EDIT ('F-VALUE FOR ANALYSIS OF VARIANCE...',ANS(7))
         (SKIP(1),COLUMN(12),A,F(9,3)),.
   PUT EDIT ('STANDARD ERROR OF ESTIMATE.........',ANS(8))
         (SKIP(1),COLUMN(12),A,F(9,3)),.
   PUT EDIT ('(ADJUSTED FOR D.F.)...........',ANS(11))
         (SKIP(1),COLUMN(17),A,F(9,3)),.
   PUT EDIT ('VARIABLE','REGRESSION','STD. ERROR OF ','COMPUTED',
         'BETA','NUMBER','COEFFICIENT','REG. COEFF.','T-VALUE',
         'COEFFICIENT')
         (SKIP(2),COLUMN(12),5(A,X(5)),SKIP(1),COLUMN(13),A,X(6),A,
         X(4),A,X(8),A,X(6),A),.
   N     =NSTEP(4),.
         DO I = 1 TO N,.
         PUT EDIT (L(I),B(I),S(I),T(I),BETA(I)) (SKIP(1),COLUMN(14),
               F(3),F(18,5),F(16,5),F(14,3),F(14,5)),.
         END,.
   PUT EDIT ('INTERCEPT',ANS(9)) (SKIP,COLUMN(12),A,F(14,5)),.
   NSTOP='0',.
   RETURN,.
   END,.                                      /*END OF PROCEDURE SOUT  */
```

CANONICAL CORRELATION CANO

Problem Description

This program analyzes the interrelations between two sets of variables measured on the same subjects. These variables are predictors in one set and criteria in the other set, but it is irrelevant whether the variables in the first set or in the second set are considered as the prediction variables. The canonical correlation, which gives the maximum correlation between linear functions of the two sets of variables, is calculated. χ^2 is also computed to test the significance of canonical correlation.

The sample problem for canonical correlation consists of four variables in the first set (left-hand side) and three variables in the second set (right-hand side) as presented in Table 2. These two sets of measurements have been made on 23 subjects.

Table 2. Sample Data for Canonical Correlation

	First set				Second set		
Observation	X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3
1	191	155	65	19	179	145	70
2	195	149	70	20	201	152	69
3	181	148	71	19	185	149	75
4	183	153	82	18	188	149	86
5	176	144	67	18	171	142	71
6	208	157	81	22	192	152	77
7	189	150	75	21	190	149	72
8	197	159	90	20	189	152	82
9	188	152	76	19	197	159	84
10	192	150	78	20	187	151	72
11	179	158	99	18	186	148	89
12	183	147	65	18	174	147	70
13	174	150	71	19	185	152	65
14	190	159	91	19	195	157	99
15	188	151	98	20	187	158	87
16	163	137	59	18	161	130	63
17	195	155	85	20	183	158	81
18	196	153	80	21	173	148	74
19	181	145	77	20	182	146	70
20	175	140	70	19	165	137	81
21	192	154	69	20	185	152	63
22	174	143	79	20	178	147	73
23	176	139	70	20	176	143	69

Program

Description

The canonical correlation program consists of the main routine named CANO, a special input routine, DAT2, and five subroutines from the Scientific Subroutine Package: CORR, CANC, MINV, MGDU, and MSDU.

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. The number of variables in the first set (that is, left-hand variables) must be greater than or equal to the number of variables in the second set (that is, right-hand variables).
2. Up to 99,999 observations
3. Up to ten data cards per observation
4. (12 F (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 the input data cards are prepared using a different format, the input format in the special input subroutine, DAT2, must be modified. The general rules for program modification are described later.

Input

Control Card

One control card is required for each problem and is read by the main program, CANO. This card is prepared as follows:

Columns	Contents	For Sample Problem
1-6	Problem number (may be alphameric)	SAMPLE
7-11	Number of observations	00023
12-13	Number of variables in the first set (that is, left-hand variables)*	04
14-15	Number of variables in the second set (that is, right-hand variables)	03
16-17	Number of data cards per observation	01

Leading zeros do not have to be keypunched, but must be right-justified within the field.

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 (12 F (6, 0)). This format assumes twelve 6-column fields per card.

Deck Setup

Deck setup is shown in Figure 20.

*The number of variables in the first set must be greater than or equal to the number of variables in the second set.

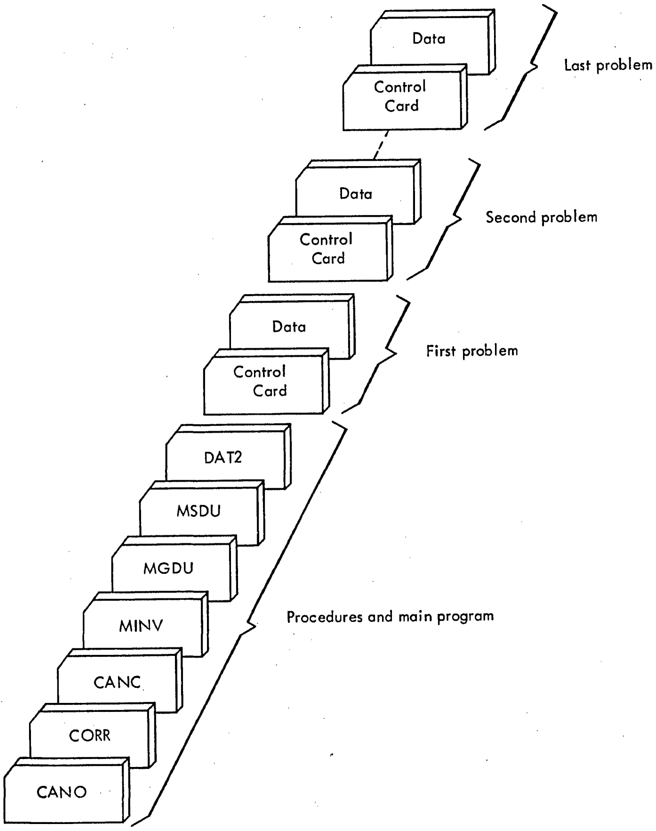
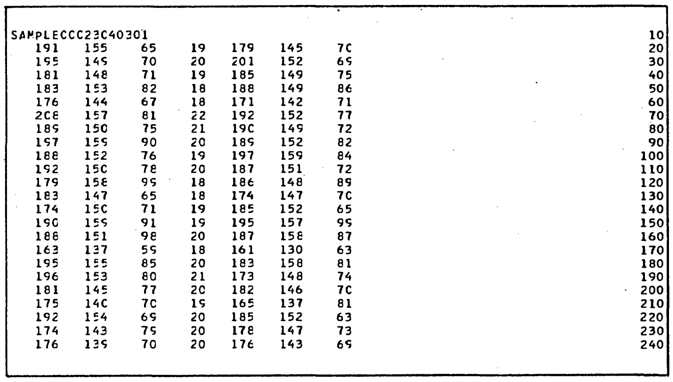


Figure 20.

Sample

The listing of input cards for the sample problem is shown in Figure 21.



```
SAMPLECCC23C40301                                                             10
   191     155     65      19     179     145     70                           20
   195     149     70      20     201     152     69                           30
   181     148     71      19     185     149     75                           40
   183     153     82      18     188     149     86                           50
   176     144     67      18     171     142     71                           60
   208     157     81      22     192     152     77                           70
   189     150     75      21     190     149     72                           80
   197     159     90      20     189     152     82                           90
   188     152     76      19     197     159     84                          100
   192     150     78      20     187     151     72                          110
   179     158     99      18     186     148     89                          120
   183     147     65      18     174     147     70                          130
   174     150     71      19     185     152     65                          140
   190     159     91      19     195     157     99                          150
   188     151     98      20     187     158     87                          160
   163     137     59      18     161     130     63                          170
   195     155     85      20     183     158     81                          180
   196     153     80      21     173     148     74                          190
   181     145     77      20     182     146     70                          200
   175     140     70      19     165     137     81                          210
   192     154     69      20     185     152     63                          220
   174     143     79      20     178     147     73                          230
   176     139     70      20     176     143     69                          240
```

Figure 21.

Output

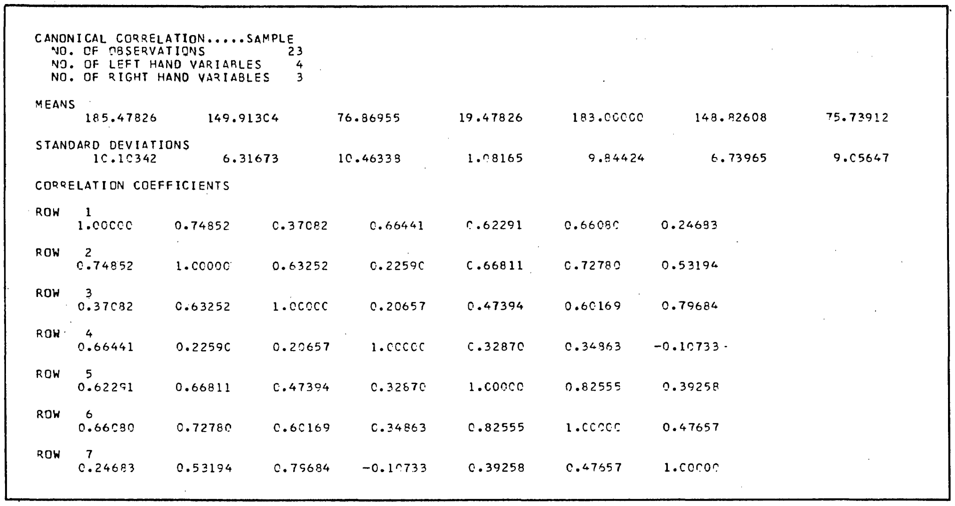
Description

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 for left- and right-hand variables.

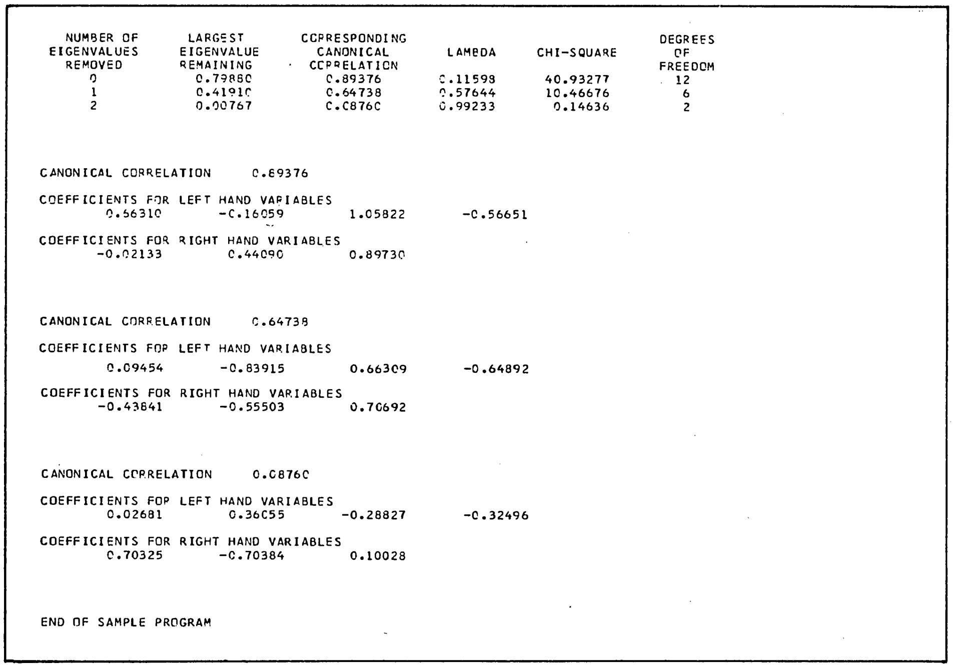
Sample

The output listing for the sample problem is shown in Figure 22.



```
CANONICAL CORRELATION.....SAMPLE
  NO. OF OBSERVATIONS          23
  NO. OF LEFT HAND VARIABLES    4
  NO. OF RIGHT HAND VARIABLES   3

MEANS
      185.47826      149.91304      76.86955      19.47826      183.00000      148.82608      75.73912

STANDARD DEVIATIONS
       10.10342        6.31673      10.46338       1.08165        9.84424        6.73965       9.05647

CORRELATION COEFFICIENTS

ROW    1
     1.00000     0.74852     0.37082     0.66441     0.62291     0.66080     0.24683

ROW    2
     0.74852     1.00000     0.63252     0.22590     0.66811     0.72780     0.53194

ROW    3
     0.37082     0.63252     1.00000     0.20657     0.47394     0.60169     0.79684

ROW    4
     0.66441     0.22590     0.20657     1.00000     0.32870     0.34863    -0.10733

ROW    5
     0.62291     0.66811     0.47394     0.32870     1.00000     0.82555     0.39258

ROW    6
     0.66080     0.72780     0.60169     0.34863     0.82555     1.00000     0.47657

ROW    7
     0.24683     0.53194     0.79684    -0.10733     0.39258     0.47657     1.00000
```

Figure 22.



```
   NUMBER OF       LARGEST        CORRESPONDING                                DEGREES
  EIGENVALUES     EIGENVALUE        CANONICAL         LAMBDA      CHI-SQUARE      OF
    REMOVED       REMAINING       CORRELATION                                  FREEDOM
       0           0.79880          0.89376          0.11598       40.93277      12
       1           0.41910          0.64738          0.57644       10.46676       6
       2           0.00767          0.08760          0.99233        0.14636       2



 CANONICAL CORRELATION      0.89376

 COEFFICIENTS FOR LEFT HAND VARIABLES
        0.56310       -0.16059        1.05822        -0.56651

 COEFFICIENTS FOR RIGHT HAND VARIABLES
       -0.02133        0.44090        0.89730



 CANONICAL CORRELATION      0.64738

 COEFFICIENTS FOR LEFT HAND VARIABLES

        0.09454       -0.83915        0.66309        -0.64892

 COEFFICIENTS FOR RIGHT HAND VARIABLES
       -0.43841       -0.55503        0.70692



 CANONICAL CORRELATION      0.08760

 COEFFICIENTS FOR LEFT HAND VARIABLES
        0.02681        0.36055       -0.28827        -0.32496

 COEFFICIENTS FOR RIGHT HAND VARIABLES
        0.70325       -0.70384        0.10028



 END OF SAMPLE PROGRAM
```

Figure 22. (Continued)

Program Modifications

Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with program modifications, the following general rule is supplied in terms of the sample problem:

1. Changes in the input format statement of the special input routine, DAT2.

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 might have been keypunched in seven 3-column fields; if so, the format would be changed to (7 F (3, 0)). Note that the current input format statement will allow a maximum of twelve variables per card.

The special input routine 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.

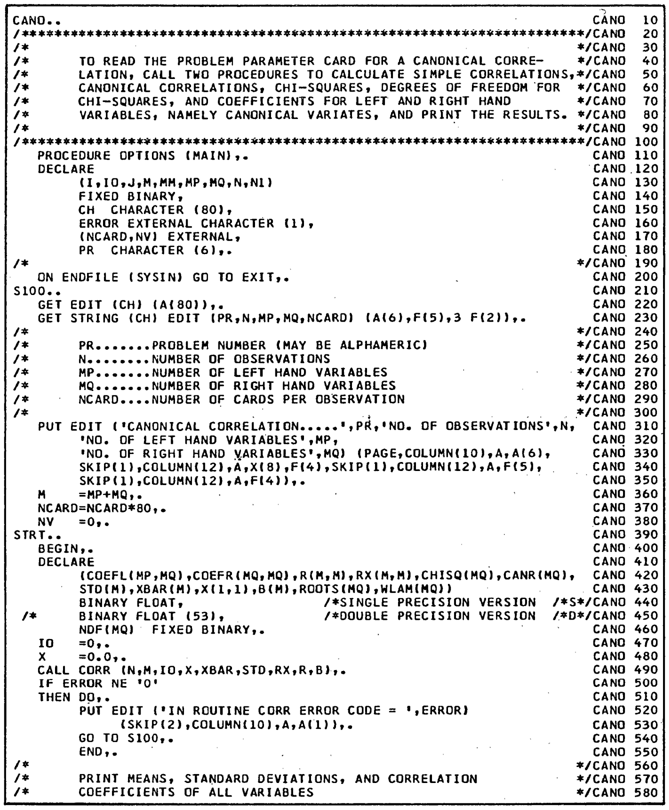
2. If there is more than one card per row of data, the value of the card count indicator (NCARD), which appears in columns 16-17 of the control card, must be changed to agree with the number of data cards per row.

Operating Instructions

The sample program for canonical correlation is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output.

Timing

The execution of this sample program on a System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 17 seconds.



```
CANO..                                                                  CANO  10
/***********************************************************************/CANO  20
/*                                                                     */CANO  30
/*      TO READ THE PROBLEM PARAMETER CARD FOR A CANONICAL CORRE-      */CANO  40
/*      LATION, CALL TWO PROCEDURES TO CALCULATE SIMPLE CORRELATIONS,*/CANO  50
/*      CANONICAL CORRELATIONS, CHI-SQUARES, DEGREES OF FREEDOM FOR  */CANO  60
/*      CHI-SQUARES, AND COEFFICIENTS FOR LEFT AND RIGHT HAND         */CANO  70
/*      VARIABLES, NAMELY CANONICAL VARIATES, AND PRINT THE RESULTS. */CANO  80
/*                                                                     */CANO  90
/***********************************************************************/CANO 100
   PROCEDURE OPTIONS (MAIN),.                                           CANO 110
   DECLARE                                                              CANO 120
        (I,IO,J,M,MM,MP,MQ,N,N1)                                        CANO 130
        FIXED BINARY,                                                   CANO 140
        CH  CHARACTER (80),                                             CANO 150
        ERROR EXTERNAL CHARACTER (1),                                   CANO 160
        (NCARD,NV) EXTERNAL,                                            CANO 170
        PR  CHARACTER (6),.                                             CANO 180
/*                                                                     */CANO 190
   ON ENDFILE (SYSIN) GO TO EXIT,.                                      CANO 200
S100..                                                                  CANO 210
   GET EDIT (CH) (A(80)),.                                              CANO 220
   GET STRING (CH) EDIT (PR,N,MP,MQ,NCARD) (A(6),F(5),3 F(2)),.         CANO 230
/*                                                                     */CANO 240
/*      PR.......PROBLEM NUMBER (MAY BE ALPHAMERIC)                    */CANO 250
/*      N........NUMBER OF OBSERVATIONS                                */CANO 260
/*      MP.......NUMBER OF LEFT HAND VARIABLES                         */CANO 270
/*      MQ.......NUMBER OF RIGHT HAND VARIABLES                        */CANO 280
/*      NCARD....NUMBER OF CARDS PER OBSERVATION                       */CANO 290
/*                                                                     */CANO 300
   PUT EDIT ('CANONICAL CORRELATION.....',PR,'NO. OF OBSERVATIONS',N,   CANO 310
        'NO. OF LEFT HAND VARIABLES',MP,                                CANO 320
        'NO. OF RIGHT HAND VARIABLES',MQ) (PAGE,COLUMN(10),A,A(6),      CANO 330
        SKIP(1),COLUMN(12),A,X(8),F(4),SKIP(1),COLUMN(12),A,F(5),       CANO 340
        SKIP(1),COLUMN(12),A,F(4)),.                                    CANO 350
   M    =MP+MQ,.                                                        CANO 360
   NCARD=NCARD*80,.                                                     CANO 370
   NV   =0,.                                                            CANO 380
STRT..                                                                  CANO 390
   BEGIN,.                                                              CANO 400
   DECLARE                                                              CANO 410
        (COEFL(MP,MQ),COEFR(MQ,MQ),R(M,M),RX(M,M),CHISQ(MQ),CANR(MQ),   CANO 420
        STD(M),XBAR(M),X(1,1),B(M),ROOTS(MQ),WLAM(MQ))                  CANO 430
        BINARY FLOAT,                   /*SINGLE PRECISION VERSION  /*S*/CANO 440
 /*     BINARY FLOAT (53),              /*DOUBLE PRECISION VERSION  /*D*/CANO 450
        NDF(MQ)  FIXED BINARY,.                                         CANO 460
   IO   =0,.                                                            CANO 470
   X    =0.0,.                                                          CANO 480
   CALL CORR (N,M,IO,X,XBAR,STD,RX,R,B),.                               CANO 490
   IF ERROR NE '0'                                                      CANO 500
   THEN DO,.                                                            CANO 510
        PUT EDIT ('IN ROUTINE CORR ERROR CODE = ',ERROR)                CANO 520
             (SKIP(2),COLUMN(10),A,A(1)),.                              CANO 530
        GO TO S100,.                                                    CANO 540
        END,.                                                           CANO 550
/*                                                                     */CANO 560
/*      PRINT MEANS, STANDARD DEVIATIONS, AND CORRELATION              */CANO 570
/*      COEFFICIENTS OF ALL VARIABLES                                  */CANO 580
/*                                                                    */
   PUT EDIT ('MEANS') (R(FM1)),.
FM1..
   FORMAT (SKIP(2),COLUMN(10),A),.
   PUT EDIT ((XBAR(I) DO I= 1 TO M)) (R(FM2)),.
FM2..
   FORMAT (SKIP,COLUMN(10),7 F(15,5)),.
   PUT EDIT ('STANDARD DEVIATIONS') (R(FM1)),.
   PUT EDIT ((STD(I) DO I= 1 TO M)) (R(FM2)),.
   PUT EDIT ('CORRELATION COEFFICIENTS') (SKIP(2),COLUMN(10),A),.
        DO I = 1 TO M,.
        PUT EDIT ('ROW',I) (SKIP(2),COLUMN(10),A,F(4)),.
        PUT EDIT ((R(I,J) DO J= 1 TO M)) (SKIP,COLUMN(10),9 F(12,5)),.
        END,.
   CALL CANC (N,MP,MQ,R,ROOTS,WLAM,CANR,CHISQ,NDF,COEFR,COEFL),.
   IF ERROR NE '0'
   THEN DO,.
        PUT EDIT ('IN ROUTINE CANC ERROR CODE = ',ERROR)
             (SKIP(2),COLUMN(10),A,A(1)),.
        IF ERROR = '1'
        THEN GO TO S100,.
        END,.
/*                                                                    */
/*      PRINT EIGENVALUES, CANONICAL CORRELATIONS, LAMBDA,            */
/*      CHI-SQUARES DEGREES OF FREEDOM                                */
/*                                                                    */
   PUT EDIT ('NUMBER OF ','LARGEST','CORRESPONDING','DEGREES',
        'EIGENVALUES','EIGENVALUE','CANONICAL','LAMBDA',
        'CHI-SQUARE','OF','REMOVED','REMAINING','CORRELATION',
        'FREEDOM') (SKIP(4),COLUMN(13),A,X(5),A,X(7),A,X(31),A,
        SKIP,COLUMN(11),A,X(5),A,X(7),A,X(7),A,X(5),A,X(7),A,
        SKIP,COLUMN(13),A,X(7),A,X(7),A,X(32),A),.
        DO I = 1 TO MQ,.
        N1   =I-1,.
/*                                                                    */
/*      TEST WHETHER EIGENVALUE IS GREATER THAN ZERO                  */
/*                                                                    */
        MM   =N1,.
        IF ROOTS(I) GT 0.0
        THEN DO,.
             PUT EDIT (N1,ROOTS(I);CANR(I),WLAM(I),CHISQ(I),NDF(I))
                  (SKIP(1),COLUMN(10),F(7),F(19,5),F(16,5),
                  2 F(14,5),X(5),F(5)),.
             MM   =MQ,.
             END,.
        END,.
/*                                                                    */
/*      PRINT CANONICAL CORRELATION                                   */
/*                                                                    */
        DO I = 1 TO MM,.
        PUT EDIT ('CANONICAL CORRELATION',CANR(I)) (SKIP(5),COLUMN(10),
             A,F(12,5)),.
        PUT EDIT ('COEFFICIENTS FOR LEFT HAND VARIABLES') (R(FM1)),.
        PUT EDIT ((COEFL(J,I) DO J= 1 TO MP)) (R(FM2)),.
        PUT EDIT ('COEFFICIENTS FOR RIGHT HAND VARIABLES') (R(FM1)),.
        PUT EDIT ((COEFR(J,I) DO J= 1 TO MQ)) (R(FM2)),.
        END,.
   END,.
   GO TO S100,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')
        (SKIP(5),COLUMN(10),A),.
   END,.                                 /*END OF PROCEDURE CANO      */
```

```
DAT2..
/**********************************************************************/
/*                                                                    */
/*      TO READ FLOATING POINT DATA, ONE OBSERVATION AT A TIME.       */
/*      DATA MAY BE SAVED ON A DATA SET.                              */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,D),.
   DECLARE
        XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),
        (NCARD,NV) EXTERNAL,
        CH CHARACTER(NCARD),
        (I,M,MM) FIXED BINARY,
        D(*) FLCAT BINARY,.
/*                                                                    */
   ON ENDFILE (SYSIN)
   GO TO EXIT,.
   GET EDIT (CH) (A(NCARD)),.
   MM   =CEIL(M/12),.
   GET STRING (CH) EDIT ((D(I) DO I= 1 TO M))
        ((MM)((12)F(6,0),X(8))),.
   IF NV= 1
   THEN PUT FILE (XDATA) EDIT ((D(I) DO I= 1 TO M)) ((M)F(6,C)),.
   REVERT ENDFILE (SYSIN),.
   RETURN,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('ERROR  INSUFFICIENT DATA')
        (SKIP(1),COLUMN(10),A),.
   STOP,.
   END,.                                 /*END OF PROCEDURE DAT2      */
```

ANALYSIS OF VARIANCE ANOV

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 first reducing them 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 presented in Table 3. 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 3. Sample Data for Analysis of Variance

Replicate (Block)		b_1				b_2				b_3			
		a_1	a_2	a_3	a_4	a_1	a_2	a_3	a_4	a_1	a_2	a_3	a_4
r_1	c_1	3	10	9	8	24	8	9	3	2	8	9	8
	c_2	4	12	3	9	22	7	16	2	2	2	7	2
	c_3	5	10	5	8	23	9	17	3	2	8	6	3
r_2	c_1	2	14	9	13	29	16	11	3	2	7	5	3
	c_2	7	11	5	8	28	18	10	6	6	6	5	9
	c_3	9	10	27	8	28	16	11	7	8	9	8	15

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 3 may be represented in the form:

$$x_{abcr} \quad \text{where} \quad a = 1,2,3,4$$

$$b = 1,2,3$$

$$c = 1,2,3$$

$$r = 1,2$$

The general principle of the analysis-of-variance procedure used in the program is first to perform a formal factorial analysis and then to 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 program consists of the main routine, named ANOV, a special input routine DAT3, and one subroutine from the Scientific Subroutine Package: AVAR.

*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.

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 14 factors
2. The total number of data points is limited only by the size of available core storage used for input.
3. (12 F(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 the input data cards are prepared using a different format, the input format statement must be modified. The general rules for program modifications are described later.

Input

Control Cards

Two control cards are required for each problem and are read by the main program, ANOV.

The first card is prepared as follows:

Columns	Contents	For Sample Problem
1-6	Problem number (may be alphameric)	SAMPLE
7-8	Number of factors	04

The second card is prepared as follows:

Columns	Contents	For Sample Problem
1	Label for the first factor	A
2-5	Number of levels for the first factor	0004
6	Label for the second factor	B
7-10	Number of levels for the second factor	0003
11	Label for the third factor	C
12-15	Number of levels for the third factor	0003
16	Label for the fourth factor	R
17-20	Number of levels for the fourth factor	0002
	•	
	•	
	•	
66	Label of the fourteenth factor	
67-70	Number of levels of the fourteenth factor	

Leading zeros do not have to be keypunched.

Data Cards

Data is keypunched in the following order: X_{1111}, X_{2111}, X_{3111}, X_{1211}, X_{2211}, X_{3211}, ..., X_{4332}. In other words, the leftmost subscript (namely, the first factor) is changed first; then the second, third, and fourth subscripts. In the sample problem, the first subscript corresponds to factor A; the second, third, and fourth subscripts, to factors B, C, and R. Since the number of data fields per card is twelve, implied by the format (12 F(6, 0)), each row in Table 3 is keypunched on a separate card.

Deck Setup

Deck setup is shown in Figure 23.

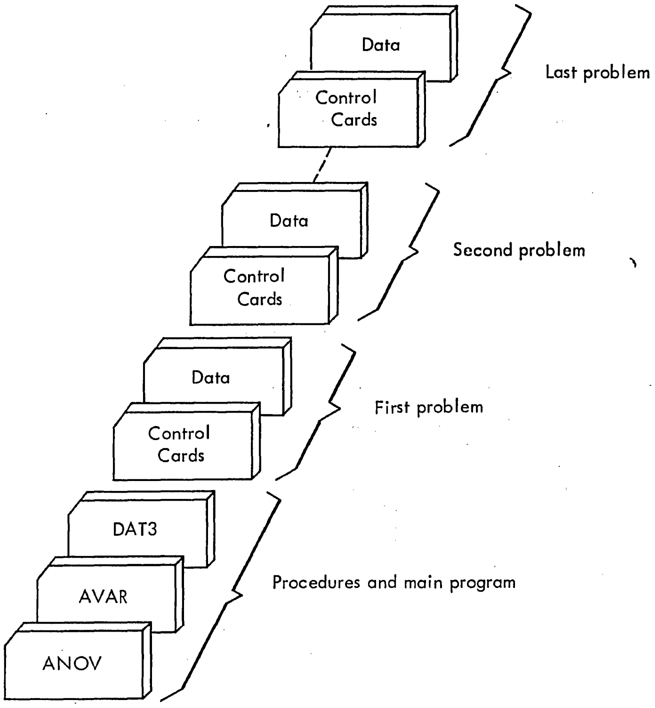
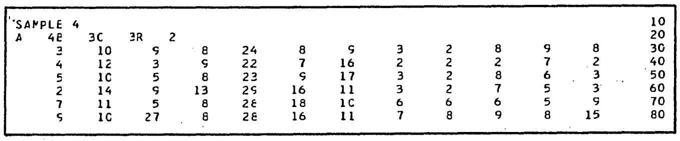


Figure 23.

Sample

The listing of input cards for the sample problem is shown in Figure 24.



```
SAMPLE 4                                                                       10
A   4B   3C   3R   2                                                           20
     3    10     9     8    24     8     9     3     2     8     9     8        30
     4    12     3     9    22     7    16     2     2     2     7     2        40
     5    10     5     8    23     9    17     3     2     8     6     3        50
     2    14     9    13    29    16    11     3     2     7     5     3        60
     7    11     5     8    28    18    10     6     6     6     5     9        70
     9    10    27     8    28    16    11     7     8     9     8    15        80
```

Figure 24.

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. Some of these are presented in Table 4.

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 25. 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 25. Since these data are purely hypothetical, interpretations of the various effects are not made.

Table 4. Instructions to Summarize Components of Analysis of Variance

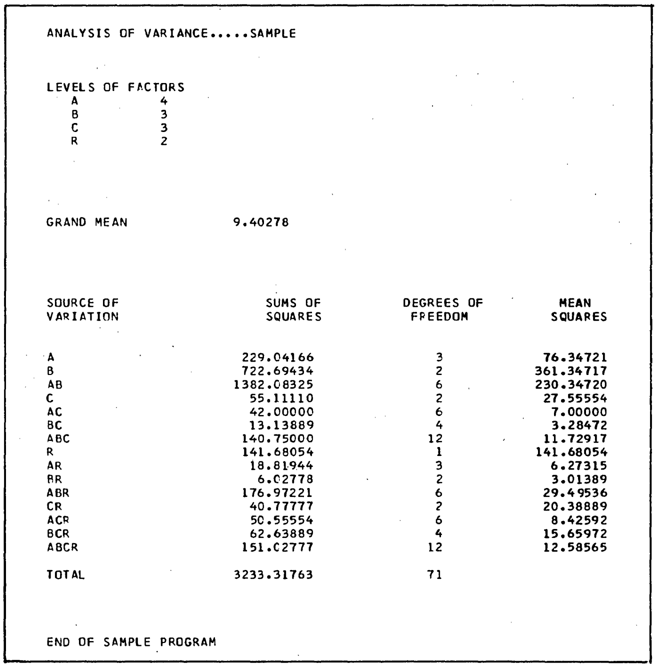
	Single Classification with Replicates	Two-way Classification with Cell Replicates	Randomized Complete Block with Two Factors	Split Plot

(Input)
Factor No. 1
2
3
	
	Groups = A
	Replicates = R
		
		Rows = A
		Columns = B
		Replicates = R
			
			Factor 1 = A
			Factor 2 = B
			Blocks = R
				
				Main treatment = A
				Subtreatment = B
				Blocks = R

(Output) Sums of squares	 A R AR	 A B AB R AR BR ABR	 A B AB R AR BR ABR	 A B AB R AR BR ABR
Summary instruction	Error = R + (AR)	Error = R + (AR) +(BR)+(ABR)	Error = (AR)+(BR) +(ABR)	Error = (BR)+(ABR) (b)
Analysis of variance	Groups A Error	Rows A Columns B Interaction AB Error	Factor 1 A Factor 2 B Interaction AB Blocks R Error	Main treatment A Blocks R Error (a) AR Subtreatment B Interaction AB Error (b)

Sample

The output listing for the sample problem is shown in Figure 25.



```
ANALYSIS OF VARIANCE.....SAMPLE


LEVELS OF FACTORS
   A          4
   B          3
   C          3
   R          2




GRAND MEAN              9.40278




SOURCE OF                   SUMS OF          DEGREES OF          MEAN
VARIATION                   SQUARES           FREEDOM           SQUARES


A                         229.04166              3             76.34721
B                         722.69434              2            361.34717
AB                       1382.08325              6            230.34720
C                          55.11110              2             27.55554
AC                         42.00000              6              7.00000
BC                         13.13889              4              3.28472
ABC                       140.75000             12             11.72917
R                         141.68054              1            141.68054
AR                         18.81944              3              6.27315
BR                          6.02778              2              3.01389
ABR                       176.97221              6             29.49536
CR                         40.77777              2             20.38889
ACR                        50.55554              6              8.42592
BCR                        62.63889              4             15.65972
ABCR                      151.02777             12             12.58565

TOTAL                    3233.31763             71



END OF SAMPLE PROGRAM
```

Figure 25.

Program Modifications

Input data in a different format can also be handled by providing a different format statement. In order to familiarize the user with the program modifications, the following general rule is supplied in terms of the sample problem:

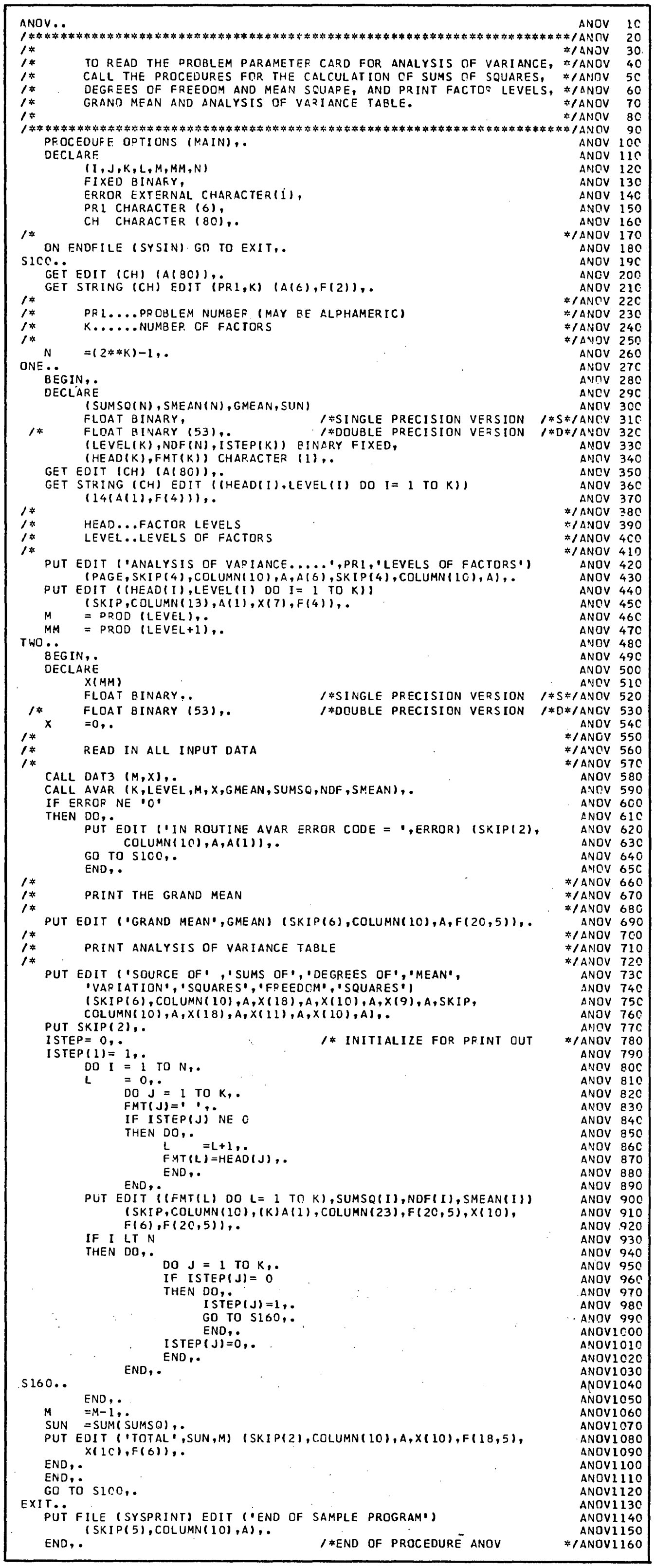
Only the format statement and the variable per card count indicator for input data may be changed. Since sample data are either one- or two-digit numbers, rather than using a six-column field, as in the sample problem, each row of data might have been keypunched in a two-column field; if so, the format is changed to (12 F(2,0)). This format assumes twelve 2-column fields per card, beginning in column 1.

Operating Instructions

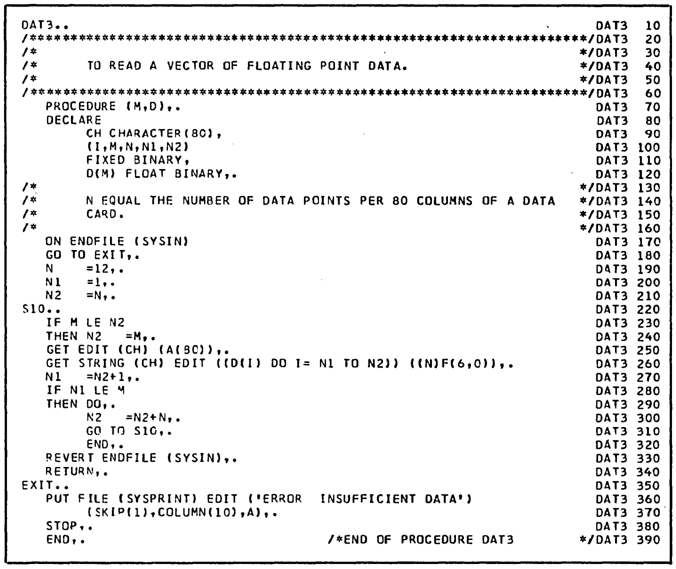
The sample analysis of variance program is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output.

Timing

The execution of this sample program on a System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 11 seconds.



```
ANOV..
/**********************************************************************/
/*                                                                    */
/*      TO READ THE PROBLEM PARAMETER CARD FOR ANALYSIS OF VARIANCE,  */
/*      CALL THE PROCEDURES FOR THE CALCULATION OF SUMS OF SQUARES,   */
/*      DEGREES OF FREEDOM AND MEAN SQUARE, AND PRINT FACTOR LEVELS,  */
/*      GRAND MEAN AND ANALYSIS OF VARIANCE TABLE.                    */
/*                                                                    */
/**********************************************************************/
   PROCEDURE OPTIONS (MAIN),.
   DECLARE
        (I,J,K,L,M,MM,N)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        PR1 CHARACTER (6),
        CH  CHARACTER (80),.
/*                                                                    */
   ON ENDFILE (SYSIN) GO TO EXIT,.
S100..
   GET EDIT (CH) (A(80)),.
   GET STRING (CH) EDIT (PR1,K) (A(6),F(2)),.
/*                                                                    */
/*      PR1....PROBLEM NUMBER (MAY BE ALPHAMERIC)                     */
/*      K......NUMBER OF FACTORS                                      */
/*                                                                    */
   N    =(2**K)-1,.
ONE..
   BEGIN,.
   DECLARE
        (SUMSQ(N),SMEAN(N),GMEAN,SUN)
        FLOAT BINARY,                 /*SINGLE PRECISION VERSION  /*S*/
 /*     FLOAT BINARY (53),.           /*DOUBLE PRECISION VERSION  /*D*/
        (LEVEL(K),NDF(N),ISTEP(K)) BINARY FIXED,
        (HEAD(K),FMT(K)) CHARACTER (1),.
   GET EDIT (CH) (A(80)),.
   GET STRING (CH) EDIT ((HEAD(I),LEVEL(I) DO I= 1 TO K))
        (14(A(1),F(4))),.
/*                                                                    */
/*      HEAD...FACTOR LEVELS                                          */
/*      LEVEL..LEVELS OF FACTORS                                      */
/*                                                                    */
   PUT EDIT ('ANALYSIS OF VARIANCE.....',PR1,'LEVELS OF FACTORS')
        (PAGE,SKIP(4),COLUMN(10),A,A(6),SKIP(4),COLUMN(10),A),.
   PUT EDIT ((HEAD(I),LEVEL(I) DO I= 1 TO K))
        (SKIP,COLUMN(13),A(1),X(7),F(4)),.
   M    = PROD (LEVEL),.
   MM   = PROD (LEVEL+1),.
TWO..
   BEGIN,.
   DECLARE
        X(MM)
        FLOAT BINARY,.                /*SINGLE PRECISION VERSION  /*S*/
 /*     FLOAT BINARY (53),.           /*DOUBLE PRECISION VERSION  /*D*/
   X    =0,.
/*                                                                    */
/*      READ IN ALL INPUT DATA                                        */
/*                                                                    */
   CALL DAT3 (M,X),.
   CALL AVAR (K,LEVEL,M,X,GMEAN,SUMSQ,NDF,SMEAN),.
   IF ERROR NE '0'
   THEN DO,.
        PUT EDIT ('IN ROUTINE AVAR ERROR CODE = ',ERROR) (SKIP(2),
             COLUMN(10),A,A(1)),.
        GO TO S100,.
        END,.
/*                                                                    */
/*      PRINT THE GRAND MEAN                                          */
/*                                                                    */
   PUT EDIT ('GRAND MEAN',GMEAN) (SKIP(6),COLUMN(10),A,F(20,5)),.
/*                                                                    */
/*      PRINT ANALYSIS OF VARIANCE TABLE                              */
/*                                                                    */
   PUT EDIT ('SOURCE OF' ,'SUMS OF','DEGREES OF','MEAN',
        'VARIATION','SQUARES','FREEDOM','SQUARES')
        (SKIP(6),COLUMN(10),A,X(18),A,X(10),A,X(9),A,SKIP,
        COLUMN(10),A,X(18),A,X(11),A,X(10),A),.
   PUT SKIP(2),.
   ISTEP= 0,.                         /* INITIALIZE FOR PRINT OUT     */
   ISTEP(1)= 1,.
        DO I = 1 TO N,.
        L    = 0,.
             DO J = 1 TO K,.
             FMT(J)=' ',.
             IF ISTEP(J) NE 0
             THEN DO,.
                  L    =L+1,.
                  FMT(L)=HEAD(J),.
                  END,.
             END,.
        PUT EDIT ((FMT(L) DO L= 1 TO K),SUMSQ(I),NDF(I),SMEAN(I))
             (SKIP,COLUMN(10),(K)A(1),COLUMN(23),F(20,5),X(10),
             F(6),F(20,5)),.
        IF I LT N
        THEN DO,.
                  DO J = 1 TO K,.
                  IF ISTEP(J)= 0
                  THEN DO,.
                       ISTEP(J)=1,.
                       GO TO S160,.
                       END,.
                  ISTEP(J)=0,.
                  END,.
             END,.
S160..
        END,.
   M    =M-1,.
   SUN  =SUM(SUMSQ),.
   PUT EDIT ('TOTAL',SUN,M) (SKIP(2),COLUMN(10),A,X(10),F(18,5),
        X(10),F(6)),.
   END,.
   END,.
   GO TO S100,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')
        (SKIP(5),COLUMN(10),A),.
   END,.                              /*END OF PROCEDURE ANOV         */
```



```
DAT3..
/**********************************************************************/
/*                                                                    */
/*      TO READ A VECTOR OF FLOATING POINT DATA.                      */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,D),.
   DECLARE
        CH CHARACTER(80),
        (I,M,N,N1,N2)
        FIXED BINARY,
        D(M) FLOAT BINARY,.
/*                                                                    */
/*      N EQUAL THE NUMBER OF DATA POINTS PER 80 COLUMNS OF A DATA    */
/*      CARD.                                                         */
/*                                                                    */
   ON ENDFILE (SYSIN)
   GO TO EXIT,.
   N    =12,.
   N1   =1,.
   N2   =N,.
S10..
   IF M LE N2
   THEN N2   =M,.
   GET EDIT (CH) (A(80)),.
   GET STRING (CH) EDIT ((D(I) DO I= N1 TO N2)) ((N)F(6,0)),.
   N1   =N2+1,.
   IF N1 LE M
   THEN DO,.
        N2   =N2+N,.
        GO TO S10,.
        END,.
   REVERT ENDFILE (SYSIN),.
   RETURN,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('ERROR  INSUFFICIENT DATA')
        (SKIP(1),COLUMN(10),A),.
   STOP,.
   END,.                         /*END OF PROCEDURE DAT3          */
```

DISCRIMINANT ANALYSIS MDSC

Problem Description

A set of linear functions is calculated from data on many groups for the purpose of classifying new individuals into one of several groups. The classification of an individual into a group is performed by evaluating each of the calculated linear functions, then finding the group for which the associated probability is largest.

The sample problem for discriminant analysis consists of four groups of observations, as presented in Table 5. The number of observations in the first group is eight, the second group seven, the third group seven, and the fourth group eight. The number of variables in all groups is six.

Program

Description

The discriminant analysis consists of the main program MDSC, a special input routine DAT2, and three subroutines from the Scientific Subroutine Package: DMTX, MINV, DSCR.

Table 5. Sample Data for Discriminant Analysis

	Observation	X_1	X_2	X_3	X_4	X_5	X_6
Group 1	1	3	10	9	8	24	8
	2	4	12	3	8	22	7
	3	9	3	2	8	9	8
	4	16	2	2	2	7	2
	5	5	10	5	8	23	9
	6	17	3	2	8	6	3
	7	2	10	9	8	29	16
	8	7	10	5	8	28	18
Group 2	1	9	10	27	8	28	16
	2	11	7	8	9	8	15
	3	8	10	2	8	27	16
	4	1	6	8	14	14	13
	5	7	8	9	6	18	2
	6	7	9	8	2	19	9
	7	7	10	5	8	27	17
Group 3	1	3	11	9	15	20	10
	2	9	4	10	7	9	9
	3	4	13	10	7	21	15
	4	8	5	16	16	16	7
	5	6	9	10	5	23	11
	6	8	10	5	8	27	16
	7	17	3	2	7	6	3
Group 4	1	3	10	8	8	23	8
	2	4	12	3	8	23	7
	3	9	3	2	8	21	7
	4	15	2	2	2	7	2
	5	9	10	26	8	27	16
	6	8	9	2	8	26	16
	7	7	8	6	9	18	2
	8	7	10	5	8	26	16

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 25 groups
2. The number of variables and the number of observations depend on the size of core available for input.
3. (12 F(6, 0)) format for input data. Therefore, if a problem satisfies the above conditions, it is not necessary to modify the sample program. However, if input data cards are prepared using a different format, the input format statement in the special input routine may be modified. The general rules for program modification are described later.

Input

Control Cards

Two control cards are required for each problem and are read by the main program, MDSC.

The first card is prepared as follows:

Columns	Contents	For Sample Problem
1-6	Problem number (may be alphameric)	SAMPLE
7-8	Number of groups (greater than 1)	04
9-10	Number of variables	06
11-12	Number of cards per observation	01

The second card is prepared as follows:

Columns	Contents	For Sample Problem
1-3	Number of observations in the first group	08
4-6	Number of observations in the second group	07
7-9	Number of observations in the third group	08
10-12	Number of observations in the fourth group	
	•	
	•	
	•	
73-75	Number of observations in the 25th group	

Leading zeros are not required to be keypunched.

Data Cards

Since input data are read into the computer one observation at a time, each row of data in Table 5 is keypunched on a separate card, using the format (12 F(6, 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.

If there is more than one data card per observation, the data card count indicator (NCARD), which appears in columns 11-12 of the first control card, must be changed to agree with the number of data cards per observation.

Deck Setup

The deck setup is shown in Figure 26.

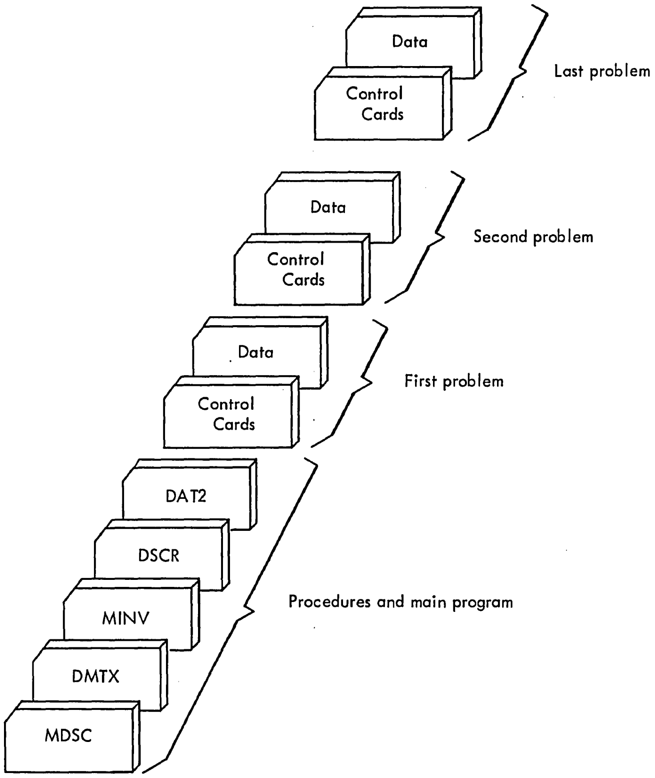
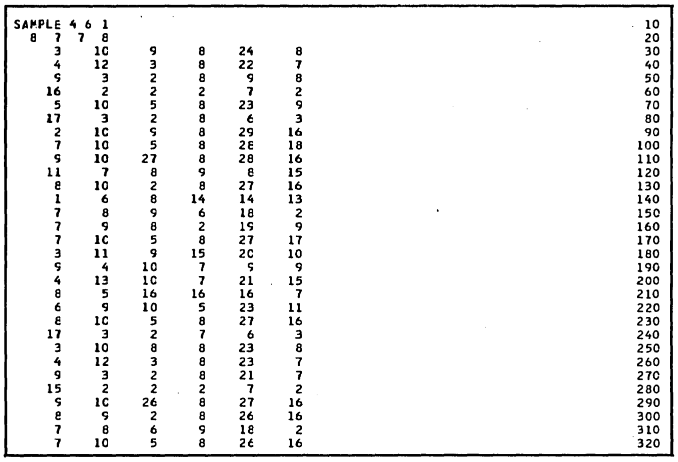


Figure 26.

Sample

The listing of input cards for the sample is shown in Figure 27.



```
SAMPLE 4 6 1                                                   10
 8  7  7  8                                                    20
     3     10      9      8     24      8                      30
     4     12      3      8     22      7                      40
     9      3      2      8      9      8                      50
    16      2      2      2      7      2                      60
     5     10      5      8     23      9                      70
    17      3      2      8      6      3                      80
     2     10      9      8     29     16                      90
     7     10      5      8     28     18                     100
     9     10     27      8     28     16                     110
    11      7      8      9      8     15                     120
     8     10      2      8     27     16                     130
     1      6      8     14     14     13                     140
     7      8      9      6     18      2                     150
     7      9      8      2     19      9                     160
     7     10      5      8     27     17                     170
     3     11      9     15     20     10                     180
     9      4     10      7      9      9                     190
     4     13     10      7     21     15                     200
     8      5     16     16     16      7                     210
     6      9     10      5     23     11                     220
     8     10      5      8     27     16                     230
    17      3      2      7      6      3                     240
     3     10      8      8     23      8                     250
     4     12      3      8     23      7                     260
     9      3      2      8     21      7                     270
    15      2      2      2      7      2                     280
     9     10     26      8     27     16                     290
     8      9      2      8     26     16                     300
     7      8      6      9     18      2                     310
     7     10      5      8     26     16                     320
```

Figure 27.

Output

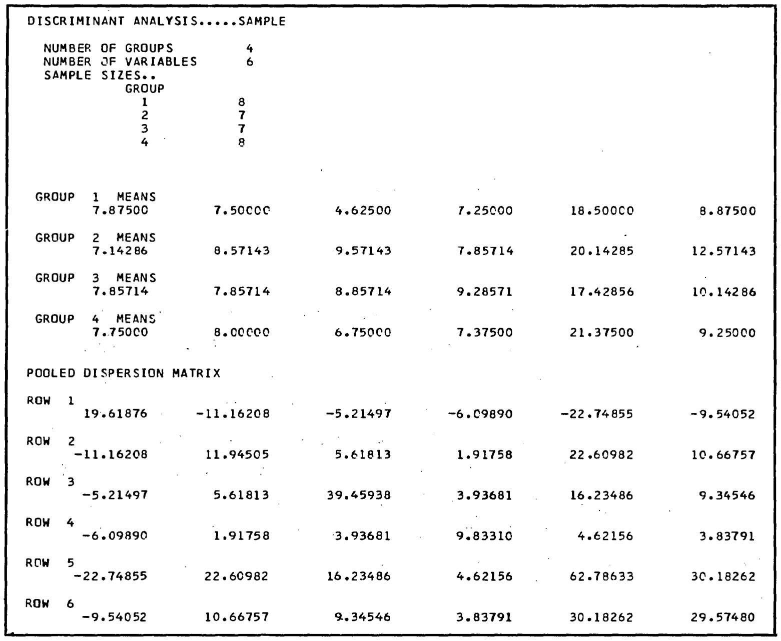
Description

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. General Mahalanobis D-square
5. Constant and coefficient of each discriminant function
6. Probability associated with the largest discriminant function evaluated for each observation

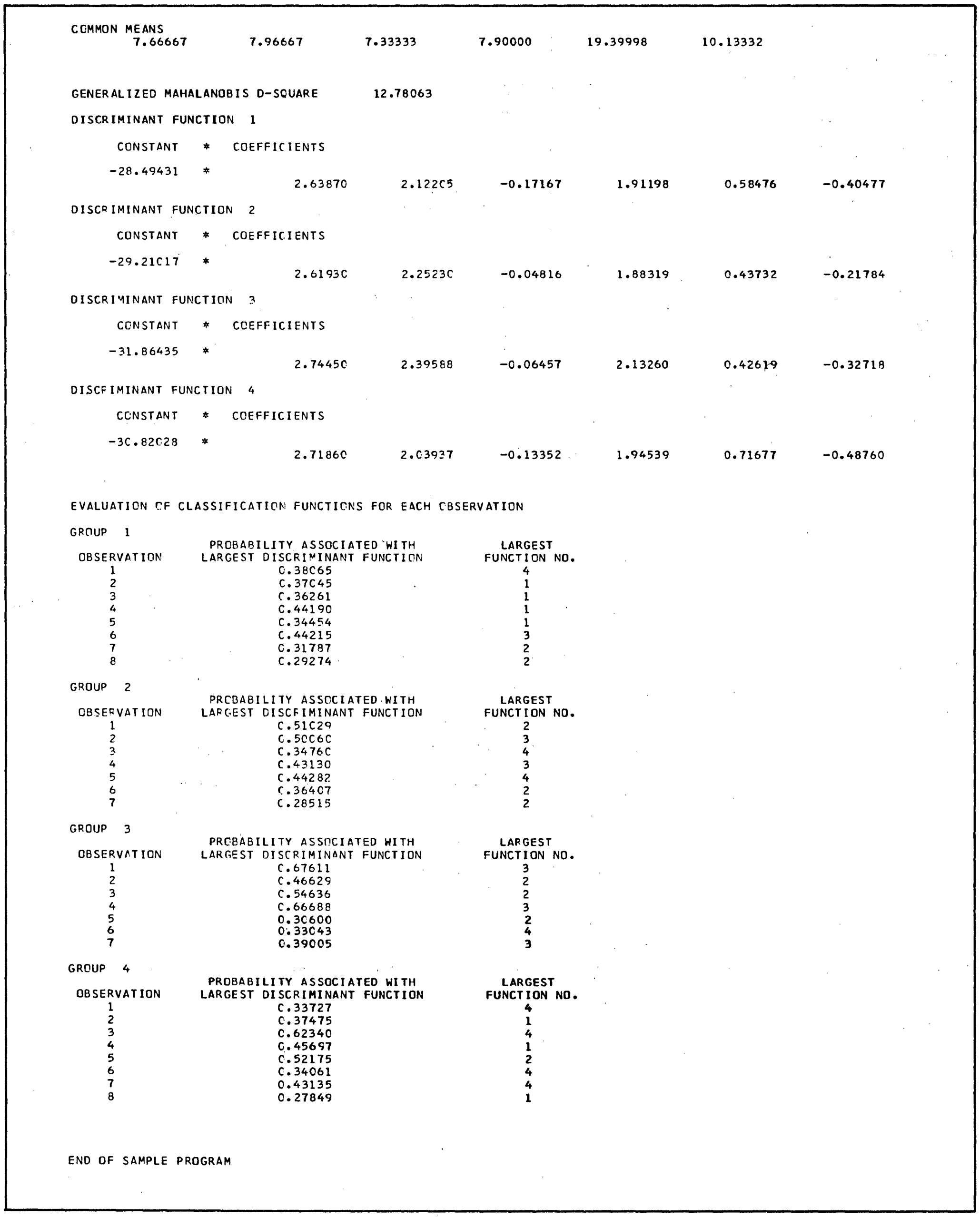
Sample

The output listing for the sample problem is shown as Figure 28.



```
DISCRIMINANT ANALYSIS.....SAMPLE

  NUMBER OF GROUPS          4
  NUMBER OF VARIABLES       6
  SAMPLE SIZES..
           GROUP
             1             8
             2             7
             3             7
             4             8


 GROUP   1  MEANS
        7.87500       7.50000       4.62500       7.25000      18.50000       8.87500

 GROUP   2  MEANS
        7.14286       8.57143       9.57143       7.85714      20.14285      12.57143

 GROUP   3  MEANS
        7.85714       7.85714       8.85714       9.28571      17.42856      10.14286

 GROUP   4  MEANS
        7.75000       8.00000       6.75000       7.37500      21.37500       9.25000

POOLED DISPERSION MATRIX

ROW   1
       19.61876     -11.16208      -5.21497      -6.09890     -22.74855      -9.54052

ROW   2
      -11.16208      11.94505       5.61813       1.91758      22.60982      10.66757

ROW   3
       -5.21497       5.61813      39.45938       3.93681      16.23486       9.34546

ROW   4
       -6.09890       1.91758       3.93681       9.83310       4.62156       3.83791

ROW   5
      -22.74855      22.60982      16.23486       4.62156      62.78633      30.18262

ROW   6
       -9.54052      10.66757       9.34546       3.83791      30.18262      29.57480
```

Figure 28.



```
CGMMON MEANS
       7.66667         7.96667         7.33333         7.90000         19.39998        10.13332


GENERALIZED MAHALANOBIS D-SQUARE        12.78063

DISCRIMINANT FUNCTION  1

       CONSTANT    *   COEFFICIENTS

      -28.49431    *
                              2.63870         2.122C5         -0.17167         1.91198         0.58476         -0.40477

DISCRIMINANT FUNCTION  2

       CONSTANT    *   COEFFICIENTS

      -29.21C17    *
                              2.6193C         2.2523C         -0.04816         1.88319         0.43732         -0.21784

DISCRIMINANT FUNCTION  3

       CCNSTANT    *   CCEFFICIENTS

      -31.86435    *
                              2.7445C         2.39588         -0.06457         2.13260         0.42619         -0.32718

DISCFIMINANT FUNCTION  4

       CCNSTANT    *   COEFFICIENTS

      -3C.82C28    *
                              2.7186C         2.C3937         -0.13352         1.94539         0.71677         -0.48760


EVALUATION CF CLASSIFICATION FUNCTICNS FOR EACH CBSERVATION

GROUP  1
                 PROBABILITY ASSOCIATED WITH              LARGEST
 OBSERVATION    LARGEST DISCRIMINANT FUNCTION           FUNCTION NO.
     1                   C.38C65                              4
     2                   C.37C45                              1
     3                   C.36261                              1
     4                   C.44190                              1
     5                   C.34454                              1
     6                   C.44215                              3
     7                   C.31787                              2
     8                   C.29274                              2

GROUP  2
                 PRCBABILITY ASSOCIATED WITH              LARGEST
 OBSERVATION    LARGEST DISCFIMINANT FUNCTION           FUNCTION NO.
     1                   C.51C29                              2
     2                   C.50C6C                              3
     3                   C.3476C                              4
     4                   C.43130                              3
     5                   C.44282                              4
     6                   C.364C7                              2
     7                   C.28515                              2

GROUP  3
                 PRCBABILITY ASSOCIATED WITH              LARGEST
 OBSERVATION    LARGEST DISCRIMINANT FUNCTION           FUNCTION NO.
     1                   C.67611                              3
     2                   C.46629                              2
     3                   C.54636                              2
     4                   C.66688                              3
     5                   0.3C600                              2
     6                   0.33C43                              4
     7                   0.39005                              3

GROUP  4
                 PROBABILITY ASSOCIATED WITH              LARGEST
 OBSERVATION    LARGEST DISCRIMINANT FUNCTION           FUNCTION NO.
     1                   C.33727                              4
     2                   C.37475                              1
     3                   C.62340                              4
     4                   C.45697                              1
     5                   C.52175                              2
     6                   C.34061                              4
     7                   0.43135                              4
     8                   0.27849                              1


END OF SAMPLE PROGRAM
```

Figure 28. (Continued)

Program Modification

1. Changes in the input format statement of the special input routine, DAT2:
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 might have been keypunched in two-column fields; if so, the format is changed to (6 F(2,0)). This format assumes six 2-column fields per card, beginning in column 1.

2. If there are more than twelve variables in a problem, each row of data is continued on the second card until the last data point is keypunched. However, each row of data must begin in a new card. If there is more than one data card per observation, the value of the data card count indicator (NCARD), which appears in columns 11-12 of the first control card, must be changed to agree with the number of data cards.

Operating Instructions

The sample program for discriminant analysis is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output.

Timing

The execution of this sample program on a System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 28 seconds.

```
MDSC..                                                                  MDSC  10
/**********************************************************************/MDSC  20
/*                                                                    */MDSC  30
/*      TO READ THE PROBLEM PARAMETER CARD AND DATA FOR DISCRIMINANT */MDSC  40
/*      ANALYSIS, CALL THE PROCEDURES TO CALCULATE VARIABLE MEANS IN */MDSC  50
/*      EACH GROUP, POOLED DISPERSION MATRIX, COMMON COEFFICIENTS OF */MDSC  60
/*      DISCRIMINANT FUNCTIONS AND PROBABILITY ASSOCIATED WITH LARG- */MDSC  70
/*      EST DISCRIMINANT FUNCTION OF EACH CASE IN EACH GROUP, AND    */MDSC  80
/*      PRINT THE RESULTS.                                           */MDSC  90
/*                                                                    */MDSC 100
/**********************************************************************/MDSC 110
   PROCEDURE OPTIONS (MAIN),.                                           MDSC 120
   DECLARE                                                              MDSC 130
        (I,J,K,L,M,N1,N2,NN)                                            MDSC 140
        FIXED BINARY,                                                   MDSC 150
        PR1 CHARACTER (6),                                              MDSC 160
        ERROR EXTERNAL CHARACTER (1),                                   MDSC 170
        (NCARD,NV) EXTERNAL,                                            MDSC 180
        CH  CHARACTER (80),.                                            MDSC 190
/*                                                                    */MDSC 200
   ON ENDFILE (SYSIN) GO TO EXIT,.                                      MDSC 210
S100..                                                                  MDSC 220
   GET EDIT (CH) (A(80)),.                                              MDSC 230
   GET STRING (CH) EDIT (PR1,K,M,NCARD) (A(6),3 F(2)),.                 MDSC 240
/*                                                                    */MDSC 250
/*      PR1......PROBLEM NUMBER (MAY BE ALPHAMERIC)                   */MDSC 260
/*      K........NUMBER OF GROUPS                                     */MDSC 270
/*      M........NUMBER OF VARIABLES                                  */MDSC 280
/*      NCARD....NUMBER OF CARDS PER OBSERVATION                      */MDSC 290
/*                                                                    */MDSC 300
   NCARD=NCARD*80,.                                                     MDSC 310
   NV   =0,.                                                            MDSC 320
ONE..                                                                   MDSC 330
   BEGIN,.                                                              MDSC 340
   DECLARE                                                              MDSC 350
        N(K)  FIXED BINARY,                                             MDSC 360
        (XBAR(M,K),C(M+1,K),D(M,M),CMEAN(M),DET,V,CON)                  MDSC 370
        BINARY FLOAT,.                /*SINGLE PRECISION VERSION  /*S*/MDSC 380
 /*     BINARY FLOAT (53),.           /*DOUBLE PRECISION VERSION  /*D*/MDSC 390
/*                                                                    */MDSC 400
/*      READ SAMPLE SIZE OF EACH GROUP                                */MDSC 410
/*                                                                    */MDSC 420
   GET EDIT (CH) (A(80)),.                                              MDSC 430
   GET STRING (CH) EDIT ((N(I) DO I= 1 TO K))                           MDSC 440
        (25 F(3)),.                                                     MDSC 450
   NN   =SUM (N),.                                                      MDSC 460
TWO..                                                                   MDSC 470
   BEGIN,.                                                              MDSC 480
   DECLARE                                                              MDSC 490
        LG(NN)  FIXED BINARY,                                           MDSC 500
        X(NN,M) FLOAT BINARY,                                           MDSC 510
        P(NN)                                                           MDSC 520
        BINARY FLOAT,.                /*SINGLE PRECISION VERSION  /*S*/MDSC 530
 /*     BINARY FLOAT (53),.           /*DOUBLE PRECISION VERSION  /*D*/MDSC 540
   PUT EDIT ('DISCRIMINANT ANALYSIS.....',PR1, '  NUMBER OF GROUPS',K, MDSC 550
        '  NUMBER OF VARIABLES',M,'  SAMPLE SIZES..','GROUP')           MDSC 560
        (PAGE,SKIP(4),COLUMN(10),A,A,SKIP(2),COLUMN(10),A,X(7),F(3),   MDSC 570
        SKIP(1),COLUMN(10),A,F(7),SKIP(1),COLUMN(10),A,SKIP(1),         MDSC 580
        COLUMN(22),A),.                                                 MDSC 590
   PUT EDIT ((I,N(I) DO I= 1 TO K)) (SKIP(1),COLUMN(22),F(3),X(8),      MDSC 600
        F(4)),.                                                         MDSC 610
   PUT EDIT ('  ') (SKIP(2),A),.                                        MDSC 620
/*                                                                    */MDSC 630
/*      READ IN DATA. IN THE MANNER EQUIVALENT TO A 3-DIMENSIONAL     */MDSC 640
/*      ARRAY X(1,1,1),X(2,1,1),X(3,1,1),ETC. THE FIRST SUBSCRIPT     */MDSC 650
/*      IS THE CASE NUMBER,THE SECOND SUBSCRIPT IS THE VARIABLE       */MDSC 660
/*      NUMBER AND THE THIRD SUBSCRIPT IS THE GROUP NUMBER            */MDSC 670
/*                                                                    */MDSC 680
        DO I = 1 TO NN,.                                                MDSC 690
        CALL DAT2 (M,CMEAN),.                                           MDSC 700
             DO J = 1 TO M,.                                            MDSC 710
             X(I,J)=CMEAN(J),.                                          MDSC 720
             END,.                                                      MDSC 730
        END,.                                                           MDSC 740
   CALL DMTX (K,M,N,X,XBAR,D),.                                         MDSC 750
   IF ERROR NE '0'                                                      MDSC 760
   THEN DO,.                                                            MDSC 770
        PUT EDIT ('IN ROUTINE DMTX ERROR CODE = ',ERROR)                MDSC 780
             (SKIP(2),COLUMN(10),A,A(1)),.                              MDSC 790
        GO TO FIN,.                                                     MDSC 800
        END,.                                                           MDSC 810
/*                                                                    */MDSC 820
/*      PRINT MEANS AND POOLED DISPERSION MATRIX                      */MDSC 830
/*                                                                    */MDSC 840
        DO I = 1 TO K,.                                                 MDSC 850
        PUT EDIT ('GROUP',I,'MEANS') (SKIP(2),COLUMN(11),A,F(3),X(2),   MDSC 860
             A),.                                                       MDSC 870
        PUT EDIT ((XBAR(J,I) DO J= 1 TO M)) (SKIP(1),COLUMN(10),        MDSC 880
             (6)F(15,5)),.                                              MDSC 890
        END,.                                                           MDSC 900
   PUT EDIT ('POOLED DISPERSION MATRIX') (SKIP(3),COLUMN(10),A),.       MDSC 910
        DO I = 1 TO M,.                                                 MDSC 920
        PUT EDIT ('ROW',I) (SKIP(2),COLUMN(10),A,F(3)),.                MDSC 930
        PUT EDIT ((D(I,J) DO J= 1 TO M)) (SKIP,COLUMN(10),6 F(15,5)),. MDSC 940
        END,.                                                           MDSC 950
   CON  =0,.                                                            MDSC 960
```

```
   CALL MINV (D,M,DET,CON),.                                            MDSC 970
   IF ERROR NE '0'                                                      MDSC 980
   THEN DO,.                                                            MDSC 990
        PUT EDIT ('IN ROUTINE MINV ERROR CODE = ',ERROR) (SKIP(2),      MDSC1000
             COLUMN(10),A,A(1)),.                                       MDSC1010
        GO TO CONT,.                                                    MDSC1020
        END,.                                                           MDSC1030
   CALL DSCR (K,M,N,X,XBAR,D,CMEAN,V,C,P,LG),.                          MDSC1040
   IF ERROR NE '0'                                                      MDSC1050
   THEN DO,.                                                            MDSC1060
        PUT EDIT ('IN ROUTINE DSCR ERROR CODE = ',ERROR)                MDSC1070
             (SKIP(2),COLUMN(10),A,A(1)),.                              MDSC1080
        GO TO S100,.                                                    MDSC1090
        END,.                                                           MDSC1100
/*                                                                    */MDSC1110
/*      PRINT THE COMMON MEANS.                                       */MDSC1120
/*                                                                    */MDSC1130
   PUT EDIT ('COMMON MEANS') (SKIP(4),COLUMN(10),A),.                   MDSC1140
   PUT EDIT ((CMEAN(I) DO I= 1 TO M)) (SKIP,COLUMN(10),(6)F(15,5)),.   MDSC1150
/*                                                                    */MDSC1160
/*      PRINT GENERALIZED MAHALANOBIS D-SQUARE                        */MDSC1170
/*                                                                    */MDSC1180
   PUT EDIT ('GENERALIZED MAHALANOBIS D-SQUARE',V)                      MDSC1190
        (SKIP(4),COLUMN(10),A,F(15,5),SKIP(2)),.                        MDSC1200
/*                                                                    */MDSC1210
/*      PRINT CONSTANTS AND COEFFICIENTS OF DISCRIMINANT FUNCTIONS    */MDSC1220
/*                                                                    */MDSC1230
        DO I = 1 TO K,.                                                 MDSC1240
        PUT EDIT ('DISCRIMINANT FUNCTION',I,'CONSTANT    *',            MDSC1250
             'COEFFICIENTS') (SKIP(2),COLUMN(10),A,F(3),SKIP(2),        MDSC1260
             COLUMN(16),A,X(3),A),.                                     MDSC1270
        PUT EDIT (C(1,I),'   *   ') (SKIP(2),COLUMN(10),F(14,5),A),.    MDSC1280
        PUT EDIT ((C(J,I) DO J= 2 TO M+1)) (SKIP,COLUMN(32),            MDSC1290
             (6)F(14,5)),.                                              MDSC1300
        END,.                                                           MDSC1310
/*                                                                    */MDSC1320
/*      PRINT EVALUATION OF CLASSIFICATION FUNCTIONS OF EACH          */MDSC1330
/*      OBSERVATION.                                                  */MDSC1340
/*                                                                    */MDSC1350
   PUT EDIT ('EVALUATION OF CLASSIFICATION FUNCTIONS FOR EACH',         MDSC1360
        ' OBSERVATION') (SKIP(4),COLUMN(10),A,A),.                      MDSC1370
   N1   =1,.                                                            MDSC1380
   N2   =N(1),.                                                         MDSC1390
        DO I = 1 TO K,.                                                 MDSC1400
        PUT EDIT ('GROUP',I,'PROBABILITY ASSOCIATED WITH','LARGEST',    MDSC1410
             ' OBSERVATION','LARGEST DISCRIMINANT FUNCTION',            MDSC1420
             'FUNCTION NO.')                                            MDSC1430
             (SKIP(2),COLUMN(10),A,F(3),SKIP,COLUMN(28),A,X(11),A,      MDSC1440
             SKIP,COLUMN(10),A,X(5),A,X(8),A),.                         MDSC1450
        L    =0;.                                                       MDSC1460
             DO J = N1 TO N2,.                                          MDSC1470
             L    =L+1,.                                                MDSC1480
             PUT EDIT (L,P(J),LG(J)) (SKIP,COLUMN(10),F(6),X(20),F(8,5)MDSC1490
                  ,X(20),F(6)),.                                        MDSC1500
             END,.                                                      MDSC1510
        IF I = K                                                        MDSC1520
        THEN GO TO CONT,.                                               MDSC1530
        N1   =N1+N(I),.                                                 MDSC1540
        N2   =N2+N(I+1),.                                               MDSC1550
        END,.                                                           MDSC1560
CONT..                                                                  MDSC1570
   END,.                                                                MDSC1580
   END,.                                                                MDSC1590
   GO TO S100,.                                                         MDSC1600
EXIT..                                                                  MDSC1610
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')                   MDSC1620
        (SKIP(5),COLUMN(10),A),.                                        MDSC1630
FIN..                                                                   MDSC1640
   END,.                              /*END OF PROCEDURE MDSC        */MDSC1650
```

```
DAT2..                                                                  DAT2  10
/**********************************************************************/DAT2  20
/*                                                                    */DAT2  30
/*      TO READ FLOATING POINT DATA, ONE OBSERVATION AT A TIME.       */DAT2  40
/*      DATA MAY BE SAVED ON A DATA SET.                              */DAT2  50
/*                                                                    */DAT2  60
/**********************************************************************/DAT2  70
   PROCEDURE (M,D),.                                                    DAT2  80
   DECLARE                                                              DAT2  90
        XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),        DAT2 100
        (NCARD,NV) EXTERNAL,                                            DAT2 110
        CH CHARACTER(NCARD),                                            DAT2 120
        (I,M,MM) FIXED BINARY,                                          DAT2 130
        D(*) FLCAT BINARY,.                                             DAT2 140
/*                                                                    */DAT2 150
   ON ENDFILE (SYSIN)                                                   DAT2 160
   GO TO EXIT,.                                                         DAT2 170
   GET EDIT (CH) (A(NCARD)),.                                           DAT2 180
   MM   =CEIL(M/12),.                                                   DAT2 190
   GET STRING (CH) EDIT ((D(I) DO I= 1 TO M))                           DAT2 200
        ((MM)((12)F(6,0),X(8))),.                                       DAT2 210
   IF NV= 1                                                             DAT2 220
   THEN PUT FILE (XDATA) EDIT ((D(I) DO I= 1 TO M)) ((M)F(6,0)),.       DAT2 230
   REVERT ENDFILE (SYSIN),.                                             DAT2 240
   RETURN,.                                                             DAT2 250
EXIT..                                                                  DAT2 260
   PUT FILE (SYSPRINT) EDIT ('ERROR  INSUFFICIENT DATA')                DAT2 270
        (SKIP(1),COLUMN(10),A),.                                        DAT2 280
   STOP,.                                                               DAT2 290
   END,.                              /*END OF PROCEDURE DAT2        */DAT2 300
```

PRINCIPAL COMPONENTS ANALYSIS FACT

Problem Description

A principal component solution and the varimax rotation of the factor matrix are performed. Principal components 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 principal components analysis consists of 23 observations with nine variables, as presented in Table 6. 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 6. Sample Data for Principal Components Analysis

Observation	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
1	7	7	9	7	15	36	60	15	24
2	13	18	25	15	13	35	61	18	30
3	9	18	24	23	12	43	62	14	31
4	7	13	25	36	11	12	63	26	32
5	6	8	20	7	15	46	18	28	15
6	10	12	30	11	10	42	27	12	17
7	7	6	11	7	15	35	60	20	25
8	16	19	25	16	13	30	64	20	30
9	9	22	26	24	13	40	66	15	32
10	8	15	26	30	13	10	66	25	34
11	8	10	20	8	17	40	20	30	18
12	9	12	28	11	8	45	30	15	19
13	11	17	21	30	10	45	60	17	30
14	9	16	26	27	14	31	59	19	17
15	10	15	24	18	12	29	48	18	26
16	11	11	30	19	19	26	57	20	30
17	16	9	16	20	18	31	60	21	17
18	9	8	19	14	16	33	67	9	19
19	7	18	22	9	15	37	62	11	20
20	8	11	23	18	9	36	61	22	24
21	6	6	27	23	7	40	55	24	31
22	10	9	26	26	10	37	57	27	29
23	8	10	26	15	11	42	59	20	28

Program

Description

The principal components analysis sample program consists of a main routine, FACT, a special input routine named DAT2, and five subroutines from the Scientific Subroutine Package: CORR, MSDU, TRAC, LOAD, and VRMX.

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 96 variables can be read.
2. Up to 99999 observations can be read.
3. Up to eight data cards per observation can be read.
4. (12 F(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 input data cards are prepared using a different format, the input format statement in the input procedure, DAT2, must be modified. The general rules for program modification are described later.

Input

Control Card

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 correlation coefficients. Only those eigenvalues greater than or equal to this value are retained in the analysis. (A decimal point must be specified.)	0001.0
20-21	Number of data cards per observation.	01

Leading zeros do not have to be keypunched.

Data Cards

Since input data are read into the computer one observation at a time, each row of data in Table 6 is keypunched on a separate card, using the format (12 F(6,0)). 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

The deck setup is shown in Figure 29.

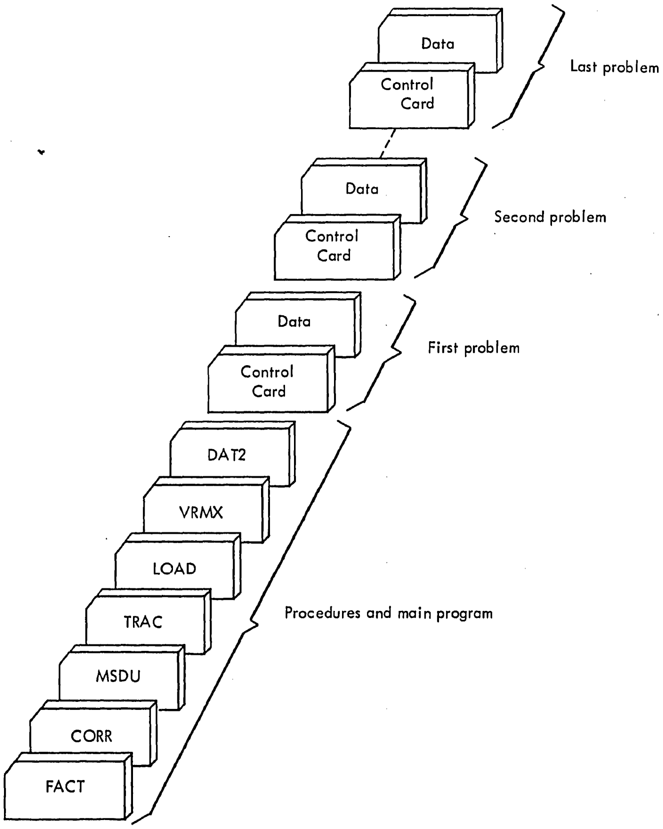
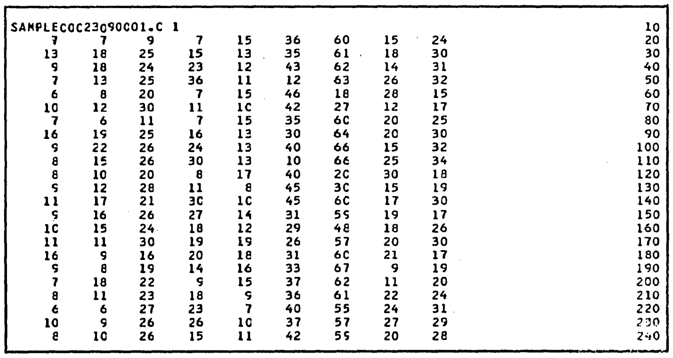


Figure 29.

Sample

The listing of input cards for the sample problem is shown in Figure 30.



```
SAMPLE00023090001.0 1                                                     10
  7     7     9     7    15    36    60    15    24                      20
 13    18    25    15    13    35    61    18    30                      30
  9    18    24    23    12    43    62    14    31                      40
  7    13    25    36    11    12    63    26    32                      50
  6     8    20     7    15    46    18    28    15                      60
 10    12    30    11    10    42    27    12    17                      70
  7     6    11     7    15    35    60    20    25                      80
 16    19    25    16    13    30    64    20    30                      90
  9    22    26    24    13    40    66    15    32                     100
  8    15    26    30    13    10    66    25    34                     110
  8    10    20     8    17    40    20    30    18                     120
  9    12    28    11     8    45    30    15    19                     130
 11    17    21    30    10    45    60    17    30                     140
  9    16    26    27    14    31    59    19    17                     150
 10    15    24    18    12    29    48    18    26                     160
 11    11    30    19    19    26    57    20    30                     170
 16     9    16    20    18    31    60    21    17                     180
  9     8    19    14    16    33    67     9    19                     190
  7    18    22     9    15    37    62    11    20                     200
  8    11    23    18     9    36    61    22    24                     210
  6     6    27    23     7    40    55    24    31                     220
 10     9    26    26    10    37    57    27    29                     230
  8    10    26    15    11    42    59    20    28                     240
```

Figure 30.

Output

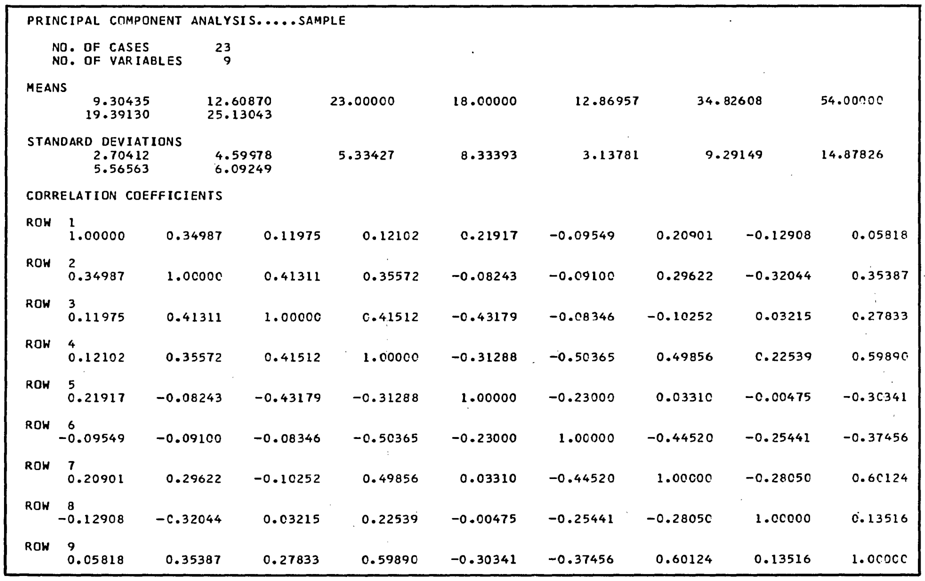
Description

The output of the sample program for principal components 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 factor matrix for each iteration cycle
9. Rotated factor matrix
10. Check on communalities

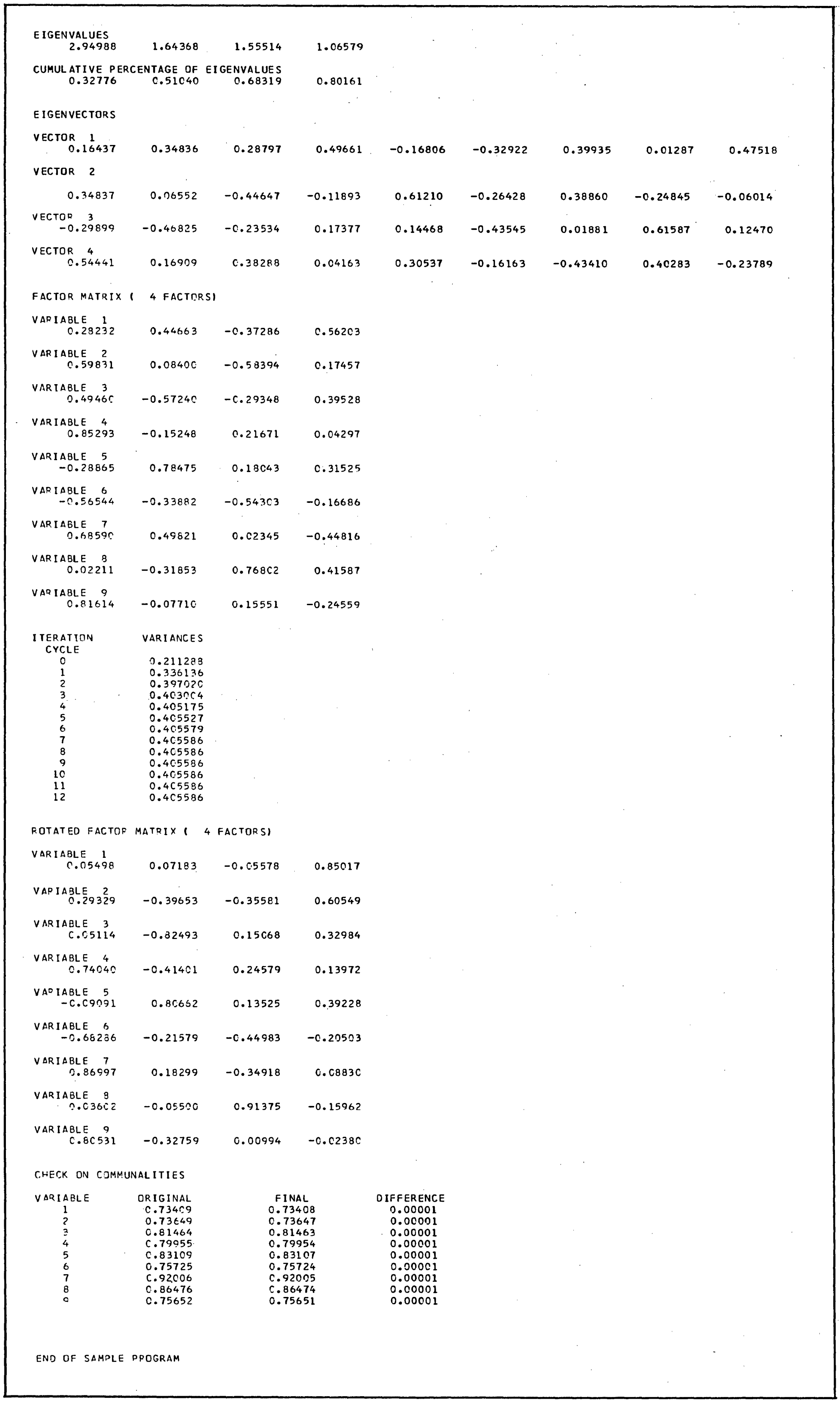
Sample

The output listing for the sample problem is shown in Figure 31.



```
PRINCIPAL COMPONENT ANALYSIS.....SAMPLE

   NO. OF CASES        23
   NO. OF VARIABLES     9

MEANS
      9.30435      12.60870       23.00000       18.00000       12.86957       34.82608       54.00000
     19.39130      25.13043

STANDARD DEVIATIONS
      2.70412       4.59978        5.33427        8.33393        3.13781        9.29149       14.87826
      5.56563       6.09249

CORRELATION COEFFICIENTS

ROW  1
     1.00000     0.34987     0.11975     0.12102     0.21917    -0.09549     0.20901    -0.12908     0.05818

ROW  2
     0.34987     1.00000     0.41311     0.35572    -0.08243    -0.09100     0.29622    -0.32044     0.35387

ROW  3
     0.11975     0.41311     1.00000     0.41512    -0.43179    -0.08346    -0.10252     0.03215     0.27833

ROW  4
     0.12102     0.35572     0.41512     1.00000    -0.31288    -0.50365     0.49856     0.22539     0.59890

ROW  5
     0.21917    -0.08243    -0.43179    -0.31288     1.00000    -0.23000     0.03310    -0.00475    -0.30341

ROW  6
    -0.09549    -0.09100    -0.08346    -0.50365    -0.23000     1.00000    -0.44520    -0.25441    -0.37456

ROW  7
     0.20901     0.29622    -0.10252     0.49856     0.03310    -0.44520     1.00000    -0.28050     0.60124

ROW  8
    -0.12908    -0.32044     0.03215     0.22539    -0.00475    -0.25441    -0.28050     1.00000     0.13516

ROW  9
     0.05818     0.35387     0.27833     0.59890    -0.30341    -0.37456     0.60124     0.13516     1.00000
```

Figure 31.



```
EIGENVALUES
     2.94988      1.64368      1.55514      1.06579

CUMULATIVE PERCENTAGE OF EIGENVALUES
     0.32776      C.51C40      0.68319      0.80161


EIGENVECTORS

VECTOR  1
     0.16437      0.34836      0.28797      0.49661     -0.16806     -0.32922      0.39935      0.01287      0.47518

VECTOR  2

     0.34837      0.06552     -0.44647     -0.11893      0.61210     -0.26428      0.38860     -0.24845     -0.06014

VECTOP  3
    -0.29899     -0.46825     -C.23534      0.17377      0.14468     -0.43545      0.01881      0.61587      0.12470

VECTOR  4
     C.54441      0.16909      C.38288      0.04163      0.30537     -0.16163     -0.43410      0.40283     -0.23789


FACTOR MATRIX (   4 FACTORS)

VAPIABLE  1
     0.28232      0.44663     -C.37286      C.56203

VARIABLE  2
     C.59831      0.0840C     -0.58394      C.17457

VARIABLE  3
     0.4946C     -0.5724C     -C.29348      0.39528

VARIABLE  4
     0.85293     -0.15248      C.21671      0.04297

VARIABLE  5
    -0.28865      0.78475      0.18C43      C.31525

VAPIABLE  6
    -C.56544     -0.33882     -0.543C3     -0.16686

VARIABLE  7
     0.6859C      0.49821      0.C2345     -0.44816

VARIABLE  8
     0.02211     -0.31853      0.768C2      0.41587

VAPIABLE  9
     0.81614     -0.0771C      0.15551     -0.24559


ITERATION       VARIANCES
  CYCLE
    0            0.211288
    1            0.336136
    2            0.39702C
    3            0.403CC4
    4            0.405175
    5            0.4C5527
    6            0.4C5579
    7            0.4C5586
    8            0.4C5586
    9            0.4C5586
   1C            0.405586
   11            0.4C5586
   12            0.4C5586


ROTATED FACTOP MATPIX (   4 FACTORS)

VARIABLE  1
     C.05498      0.07183     -0.C5578      0.85017

VAPIABLE  2
     0.29329     -0.39653     -0.35581      0.60549

VARIABLE  3
     C.C5114     -0.82493      0.15C68      0.32984

VARIABLE  4
     C.7404C     -0.414C1      0.24579      0.13972

VAPIABLE  5
    -C.C9091      0.8C662      0.13525      0.39228

VARIABLE  6
    -0.68286     -0.21579     -C.44983     -C.20503

VARIABLE  7
     0.86997      0.18299     -0.34918      C.C883C

VARIABLE  8
     0.C36C2     -0.05500      0.91375     -0.15962

VARIABLE  9
     C.8C531     -0.32759      0.00994     -0.C238C

CHECK ON COMMUNALITIES

VARIABLE        ORIGINAL              FINAL              DIFFERENCE
    1            C.734C9             0.73408              0.00001
    2            0.73649             0.73647              0.00001
    3            C.81464             0.81463              0.00001
    4            C.79955             0.79954              0.00001
    5            C.83109             0.83107              0.00001
    6            0.75725             0.75724              0.000C1
    7            C.92006             C.92005              0.00001
    8            C.86476             C.86474              0.00001
    9            C.75652             0.75651              0.00001



END OF SAMPLE PROGRAM
```

Figure 31. (Continued)

Program Modifications

Input data in a different format can also be handled by providing a different format statement. In order to familiarize the user with the program modifications, the following general rules are supplied in terms of the sample problem:

1. Changes in the input format statement of the special input subroutine, DAT2:

   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 might have been keypunched in two-column fields; if so, the format is changed to 9F(2,0). This format assumes nine 2-column fields per card, beginning in column 1.

   The special input subroutine, DAT2, is normally written by the user to handle different formats for different problems. The user may modify this procedure to perform testing of input data, transformation of data, and so on.

2. 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. If this condition exists, the value of the data card count indicator (NCARD), which appears in columns 20-21 of the control card, must be changed to agree with the number of data cards per row.

Operating Instructions

The sample program for principal components analysis is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output.

Timing

The execution of this sample program on a System/360 Model 40, using an IBM 2540 Card Reader as input and an IBM 1403, Model N1, as output, is 45 seconds.

```
FACT..                                                                  FACT  10
/**********************************************************************/FACT  20
/*                                                                    */FACT  30
/*       TO READ THE PROBLEM PARAMETER CARD, CALL FIVE PROCEDURES TO  */FACT  40
/*       PERFORM A PRINCIPAL COMPONENT SOLUTION AND THE VARIMAX ROTA- */FACT  50
/*       TION OF A FACTOR MATRIX, AND PRINT THE RESULTS.              */FACT  60
/*                                                                    */FACT  70
/**********************************************************************/FACT  80
   PROCEDURE OPTIONS (MAIN),.                                           FACT  90
   DECLARE                                                              FACT 100
        (I,IO,J,K,M,MV,N,NC,NW)                                         FACT 110
        FIXED BINARY,                                                   FACT 120
        ERROR EXTERNAL CHARACTER(1),                                    FACT 130
        (NV,NCARD) EXTERNAL,                                            FACT 140
        CON                                                             FACT 150
        FLOAT BINARY,                                                   FACT 160
        PR1 CHARACTER (6),                                              FACT 170
        CH  CHARACTER (80),.                                            FACT 180
/*                                                                    */FACT 190
   ON ENDFILE (SYSIN) GO TO EXIT,.                                      FACT 200
```

```
S10C..                                                                  FACT 210
   GET EDIT (CH) (A(80)),.                                              FACT 220
   GET STRING (CH) EDIT (PR1,N,M,CON,NCARD) (A(6),F(5),F(2),F(6,0),     FACT 230
        F(2)),.                                                         FACT 240
/*                                                                    */FACT 250
/*      PR1........PROBLEM NUMBER (MAY BE ALPHAMERIC )                */FACT 260
/*      N..........NUMBER OF CASES                                    */FACT 270
/*      M..........NUMBER OF VARIABLES                                */FACT 280
/*      CON........CONSTANT USED TO DECIDE HOW MANY EIGENVALUES       */FACT 290
/*                 TO RETAIN                                          */FACT 300
/*      NCARD......NUMBER OF DATA CARDS PER OBSERVATION               */FACT 310
/*                                                                    */FACT 320
   NCARD=NCARD*80,.                                                     FACT 330
ONE..                                                                   FACT 340
   BEGIN,.                                                              FACT 350
   DECLARE                                                              FACT 360
        (R(M,M),V(M,M),B(M),D(M),S(M),T(M),XBAR(M),TV(51),X(1,1))       FACT 370
        BINARY FLOAT,.                /*SINGLE PRECISION VERSION  /*S*/FACT 380
 /*     BINARY FLOAT (53),.           /*DOUBLE PRECISION VERSION  /*D*/FACT 390
   PUT EDIT ('PRINCIPAL COMPONENT ANALYSIS.....',PR1, 'NO. OF CASES',   FACT 400
        N,'NO. OF VARIABLES',M)                                         FACT 410
        (PAGE,SKIP(4),COLUMN(10),A,A,SKIP(2),COLUMN(13),A,X(4),F(6)     FACT 420
        ,SKIP(1),COLUMN(13),A,F(6),SKIP),.                              FACT 430
   IO   =0,.                                                            FACT 440
   X    =0,.                                                            FACT 450
   NV   =0,.                                                            FACT 460
   CALL CORR (N,M,IO,X,XBAR,S,V,R,D),.                                  FACT 470
   IF ERROR NE 'C'                                                      FACT 480
   THEN DO,.                                                            FACT 490
        PUT EDIT ('IN ROUTINE CORR ERROR CODE = ',ERROR)                FACT 500
             (SKIP(2),COLUMN(10),A,A(1)),.                              FACT 510
        GO TO S100,.                                                    FACT 520
        END,.                                                           FACT 530
   PUT EDIT ('MEANS') (SKIP(2),COLUMN(10),A),.                          FACT 540
   PUT EDIT ((XBAF(J) DO J= 1 TO M)) (SKIP,COLUMN(10),(7)F(15,5)),.     FACT 550
/*                                                                    */FACT 560
/*      PRINT MEANS AND STANDARD DEVIATIONS                           */FACT 570
/*                                                                    */FACT 580
   PUT EDIT ('STANDARD DEVIATIONS') (SKIP(2),COLUMN(10),A),.            FACT 590
   PUT EDIT ((S(J) DO J= 1 TO M)) (SKIP,COLUMN(10),(7)F(15,5)),.        FACT 600
/*                                                                    */FACT 610
/*      PRINT CORRELATION COEFFICIENTS                                */FACT 620
/*                                                                    */FACT 630
   PUT EDIT ('CORRELATION COEFFICIENTS') (SKIP(2),COLUMN(10),A),.       FACT 640
        DO I = 1 TO M,.                                                 FACT 650
        PUT EDIT ('ROW',I) (SKIP(2),COLUMN(10),A,F(3)),.                FACT 660
        PUT EDIT ((R(I,J) DO J= 1 TO M)) (SKIP,COLUMN(10),9 F(12,5)),.  FACT 670
        END,.                                                           FACT 680
   MV   =0,.                                                            FACT 690
   CALL MSDU (R,V,M,MV),.                                               FACT 700
   IF ERROR NE 'O'                                                      FACT 710
   THEN DO,.                                                            FACT 720
        PUT EDIT ('IN ROUTINE MSDU ERROR CODE ',ERROR)                  FACT 730
             (SKIP(2),COLUMN(10),A,A(1)),.                              FACT 740
        GO TO S1CC,.                                                    FACT 750
        END,.                                                           FACT 760
   CALL TRAC (M,R,CON,K,D),.                                            FACT 770
   IF ERROR NE 'C'                                                      FACT 780
   THEN DO,.                                                            FACT 790
        PUT EDIT ('IN ROUTINE TRAC ERROR CODE = ',ERROR)                FACT 800
             (SKIP(2),COLUMN(10),A,A(1)),.                              FACT 810
        GO TO S1CC,.                                                    FACT 820
        END,.                                                           FACT 830
        DO I = 1 TO K,.                 /* PRINT EIGENVALUES          */FACT 840
        S(I) =R(I,I),.                                                  FACT 850
        END,.                                                           FACT 860
   PUT EDIT ('EIGENVALUES') (SKIP(3),COLUMN(10),A),.                    FACT 870
   PUT EDIT ((S(J) DO J= 1 TO K)) (SKIP,COLUMN(10),9 F(12,5)),.         FACT 880
/*                                                                    */FACT 890
/*      PRINT CUMULATIVE PERCENTAGE OF EIGENVALUES                    */FACT 900
/*                                                                    */FACT 910
   PUT EDIT ('CUMULATIVE PERCENTAGE OF EIGENVALUES')                    FACT 920
        (SKIP(2),COLUMN(10),A),.                                        FACT 930
   PUT EDIT ((D(J) DO J= 1 TO K)) (SKIP,COLUMN(10),9 F(12,5)),.         FACT 940
/*                                                                    */FACT 950
/*      PRINT EIGENVECTORS AND FACTOR MATRIX                          */FACT 960
/*                                                                    */FACT 970
   PUT EDIT ('EIGENVECTORS') (SKIP(3),COLUMN(10),A),.                   FACT 980
        DO J = 1 TO K,.                                                 FACT 990
        PUT EDIT ('VECTOR',J) (SKIP(2),COLUMN(10),A,F(3)),.             FACT1000
        PUT EDIT ((V(I,J) DO I= 1 TO M)) (SKIP,COLUMN(10),9 F(12,5)),.  FACT1010
        END,.                                                           FACT1020
   PUT EDIT ('FACTOR MATRIX (',K,' FACTORS)')                           FACT1030
        (SKIP(3),COLUMN(10),A,F(3),A),.                                 FACT1040
   CALL LOAD (M,K,R,V),.                                                FACT1050
   IF ERROR NE 'C'                                                      FACT1060
   THEN DO,.                                                            FACT1070
        PUT EDIT ('IN ROUTINE LOAD ERROR CODE = ',ERROR)                FACT1080
             (SKIP(2),COLUMN(10),A,A(1)),.                              FACT1090
        GO TO S1CC,.                                                    FACT1100
        END,.                                                           FACT1110
        DO I = 1 TO M,.                                                 FACT1120
        PUT EDIT ('VARIABLE',I) (SKIP(2),COLUMN(10),A,F(3)),.           FACT1130
        PUT EDIT ((V(I,J) DO J= 1 TO K)) (SKIP,COLUMN(10),9 F(12,5)),.  FACT1140
        END,.                                                           FACT1150
   CALL VRMX (M,K,V,NC,TV,B,T,D),.                                      FACT1160
   IF ERROR NE 'C'                                                      FACT1170
   THEN DO,.                                                            FACT1180
        PUT EDIT ('IN ROUTINE VRMX ERROR CODE = ',ERROR)                FACT1190
             (SKIP(2),COLUMN(10),A,A(1)),.                              FACT1200
        GO TO S100,.                                                    FACT1210
        END,.                                                           FACT1220
   NW   =NC+1,.                         /* PRINT VARIANCES            */FACT1230
   PUT EDIT ('ITERATION','VARIANCES','  CYCLE') (SKIP(3),COLUMN(10),A,  FACT1240
        X(7),A,SKIP,COLUMN(10),A),.                                     FACT1250
        DO I = 1 TO NW,.                                                FACT1260
        NC   =I-1,.                                                     FACT1270
        PUT EDIT (NC,TV(I)) (SKIP,COLUMN(10),F(5),F(20,6)),.            FACT1280
        END,.                                                           FACT1290
/*                                                                    */FACT1300
/*      PRINT ROTATED FACTOR MATRIX                                   */FACT1310
/*                                                                    */FACT1320
   PUT EDIT ('ROTATED FACTOR MATRIX (',K,' FACTORS)')                   FACT1330
        (SKIP(3),COLUMN(10),A,F(3),A),.                                 FACT1340
        DO I = 1 TO M,.                                                 FACT1350
        PUT EDIT ('VARIABLE',I) (SKIP(2),COLUMN(10),A,F(3)),.           FACT1360
        PUT EDIT ((V(I,J) DO J= 1 TO K)) (SKIP,COLUMN(10),9 F(12,5)),.  FACT1370
        END,.                                                           FACT1380
/*                                                                    */FACT1390
/*      PRINT COMMUNALITIES                                           */FACT1400
/*                                                                    */FACT1410
   PUT EDIT ('CHECK ON COMMUNALITIES','VARIABLE','ORIGINAL','FINAL',    FACT1420
        'DIFFERENCE') (SKIP(3),COLUMN(10),A,SKIP(2),COLUMN(10),A,       FACT1430
        X(7),A,X(12),A,X(10),A),.                                       FACT1440
        DO I = 1 TO M,.                                                 FACT1450
```

```
         PUT EDIT (I,B(I),T(I),D(I)) (SKIP,COLUMN(10),F(5),3 F(18,5)),.  FACT1460
         END,.                                                           FACT1470
S200..                                                                   FACT1480
   END,.                                                                 FACT1490
   GO TO S100,.                                                          FACT1500
EXIT..                                                                   FACT1510
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')                    FACT1520
         (SKIP(5),COLUMN(10),A),.                                        FACT1530
   END,.                               /*END OF PROCEDURE FACT         */FACT1540
```

```
DAT2..                                                                   DAT2  10
/**********************************************************************/DAT2  20
/*                                                                    */DAT2  30
/*       TO READ FLOATING POINT DATA, ONE OBSERVATION AT A TIME.       */DAT2  40
/*       DATA MAY BE SAVED ON A DATA SET.                              */DAT2  50
/*                                                                    */DAT2  60
/**********************************************************************/DAT2  70
   PROCEDURE (M,D),.                                                     DAT2  80
   DECLARE                                                               DAT2  90
         XDATA FILE STREAM ENVIRONMENT (CONSECUTIVE V(2000,200)),        DAT2 100
         (NCARD,NV) EXTERNAL,                                            DAT2 110
         CH CHARACTER(NCARD),                                            DAT2 120
         (I,M,MM) FIXED BINARY,                                          DAT2 130
         D(*) FLOAT BINARY,.                                             DAT2 140
/*                                                                    */DAT2 150
   CN ENDFILE (SYSIN)                                                    DAT2 160
   GO TO EXIT,.                                                          DAT2 170
   GET EDIT (CH) (A(NCARD)),.                                            DAT2 180
   MM   =CEIL(M/12),.                                                    DAT2 190
   GET STRING (CH) EDIT ((D(I) DO I= 1 TO M))                            DAT2 200
         ((MM)((12)F(6,0),X(8))),.                                       DAT2 210
   IF NV= 1                                                              DAT2 220
   THEN PUT FILE (XDATA) EDIT ((D(I) DO I= 1 TO M)) ((M)F(6,0)),.        DAT2 230
   REVERT ENDFILE (SYSIN),.                                              DAT2 240
   RETURN,.                                                              DAT2 250
EXIT..                                                                   DAT2 260
  PUT FILE (SYSPRINT) EDIT ('ERROR   INSUFFICIENT DATA')                 DAT2 270
         (SKIP(1),COLUMN(10),A),.                                        DAT2 280
   STOP,.                                                                DAT2 290
   END,.                               /*END OF PROCEDURE DAT2         */DAT2 300
```

KOLMOGOROV-SMIRNOV TEST KOLM

Problem Description

This program is concerned with the problem of determining from what probability density function a particular sample is drawn, or whether two different samples were drawn from the same population. In other words, in the one-sample case, the actual distribution function of the sample is compared with one or more theoretical distribution functions, which may be normal, and/or exponential, and/or Cauchy, and/or uniform, and/or a user-specified distribution. In the two-sample case, the two sample (actual) distribution functions making up the pair are compared with one another.

From the above comparisons, a statistic is derived. In the one-sample case, this statistic evaluates the probability that the statistic will be as great as or greater than its current value, if the hypothesis that the actual (sample) and the theoretical distribution functions are equal is correct. In other words, if the probability is determined to be 0.40, for example, rejecting the hypothesis of equality of the distribution functions will be an incorrect action 40 times out of 100. Similarly, in the two-sample case, the hypothesis being tested is the equality of the two actual (sample) distribution functions.

This probability is calculated using asymptotic formulae. This means that the sample sizes involved should be large. Sizes greater than 100 are suggested by the literature. In this connection, the remarks given under subroutine SMIR should be considered.

Note also that added problems arise when, in the one-sample case, the parameters of the continuous distribution in question are estimated from the sample. Lilliefors discusses these problems (see reference given in KLMO description).

Program

Description

This program consists of the main routine KOLM, and four subroutines from the Scientific Subroutine Package: KLMO, KLM2, SMIR, and NDTR.

Capacity

This program allows up to two samples, each with 500 or fewer observations to be examined. If the user desires to modify this program to handle more observations, the instructions given below under "Program Modification" should be followed.

Input

Each job consists of two control cards and the data cards (1-3 below).

1. Job control card (minus signs in cc 1-4)
2. Program control card. Each job requires one program control card, defined below:

Columns	Contents	For Sample Problems
1 - 20	Title (alphameric)	Uniform test (Job 1) Uniform-Gauss Test (job 2)
21	1 -- one-sample test 2 -- two-sample test	1 (job 1) 2 (job 2)
22	Leave blank for one-sample test. 0 -- Read both samples (two-sample tests). 1 -- Read only one sample and compare it with the first sample read on preceding job.	0 (job 1) 1 (job 2)
23	0 -- Do not print the sample(s). 1 -- Print the sorted sample(s). (F10.3, ten per line)	0 (job 1) 1 (job 2)

(The rest of this control card pertains to a one-sample test.)

Columns	Contents	For Sample Problems
24	0 -- Do not compare with normal pdf.	1 (job 1)
	1 -- Compare with normal pdf.	
25 - 29	u -- mean of the normal pdf	0.5 (job 1)
30 - 34	s -- standard deviation of the normal pdf	0.5 (job 1)
35	0 -- Do not compare with exponential pdf.	
	1 -- Compare with exponential pdf.	1 (job 1)
36 - 40	u -- mean of the exponential pdf	0.5 (job 1)
41 - 45	s -- standard deviation of the exponential pdf	1.0 (job 1)
46	0 -- Do not compare with Cauchy pdf.	
	1 -- Compare with Cauchy pdf.	1 (job 1)
47 - 51	u -- median of the Cauchy pdf	0.5 (job 1)
52 - 56	s -- u-s is the first quartile of the Cauchy pdf	1.0 (job 1)
57	0 -- Do not compare with uniform pdf.	
	1 -- Compare with uniform pdf.	1 (job 1)
58 - 62	u -- left endpoint of the uniform pdf	0 (job 1)
63 - 67	s -- right endpoint of uniform pdf	1.0 (job 1)
68	0 -- Do not compare with user's pdf.	0 (job 1)
	1 -- Compare with user-specified pdf.	
69 - 73	u -- Parameters for	0 (job 1)
74 - 78	s user-specified pdf	0 (job 1)

u and s are described fully in "Description of Parameters" under subroutine KLMO, and are read using Format F(5,0); decimal points will override the format specification.

3. Data is read into the computer one sample at a time. The reading of a sample is terminated by a data element of 999999. New samples must begin on a new card. Data elements are punched on cards using format F(6,0), which assumes twelve 6-column fields per card; decimal points on the card override the format specification. Since the routines KLMO and KLM2 sort the samples, no particular order within a sample is necessary.

Deck Setup

The deck setup is shown in Figure 32.

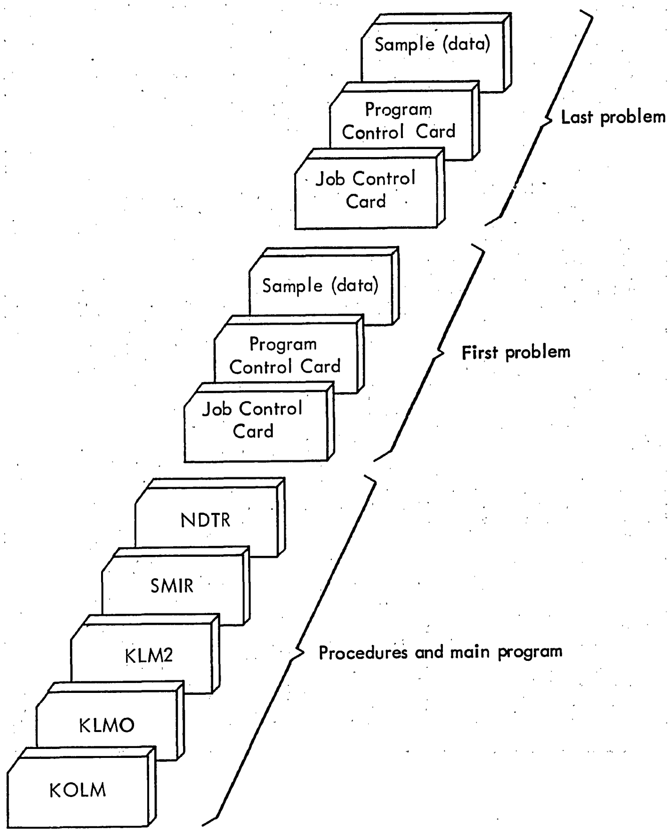
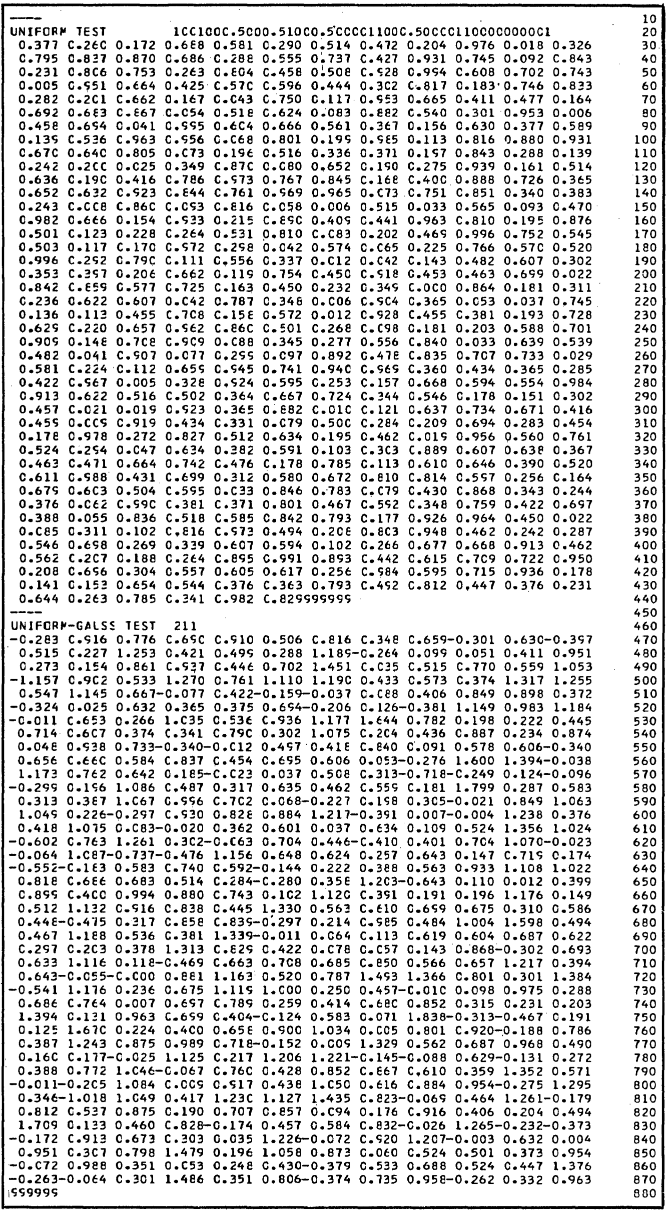


Figure 32.

Sample

The listing of input cards for the sample problems is shown in Figure 33.



```
----
UNIFORM TEST           1CC100C.5C00.510C0.5CCCC1100C.50CCC110C0C0000C1
 0.377 C.26C 0.172 0.688 0.581 0.290 0.514 0.472 0.204 0.976 0.018 0.326
 C.795 C.837 0.870 0.686 C.288 0.555 0.737 C.427 0.931 0.745 0.092 C.843
 0.231 C.8C6 0.753 0.263 C.804 C.458 0.508 C.928 0.994 C.608 0.702 0.743
 0.005 C.951 0.664 0.425 C.57C C.596 0.444 0.3C2 C.817 0.183 0.746 0.833
 0.282 C.2C1 C.662 0.167 C.C43 C.750 C.117 0.953 0.665 0.411 0.477 0.164
 0.692 0.683 C.867 C.C54 0.518 C.624 0.083 0.882 C.540 0.301 0.953 0.006
 0.458 0.694 0.041 0.995 0.6C4 0.666 0.561 0.367 0.156 0.630 0.377 0.589
 0.139 C.536 C.963 C.956 C.C68 0.801 0.199 0.985 0.113 0.816 0.880 0.931
 C.67C 0.64C 0.805 0.C73 0.196 C.516 0.336 0.371 0.197 0.843 0.288 0.139
 0.242 0.2CC 0.C25 0.349 C.87C C.C80 0.652 C.190 C.275 0.939 0.161 0.514
 0.636 C.19C 0.416 C.786 C.973 0.767 0.845 C.168 C.40C 0.888 0.726 0.365
 0.652 C.632 C.923 C.844 C.761 0.969 0.965 0.C73 C.751 C.851 0.340 0.383
 0.243 C.CC8 C.86C C.C93 C.816 C.C58 0.006 0.515 0.033 0.565 0.093 C.470
 0.982 0.666 0.154 C.933 0.215 C.89C 0.409 C.441 0.963 C.810 0.195 0.876
 0.501 C.123 0.228 C.264 0.531 0.810 C.C83 0.202 0.469 0.996 0.752 0.545
 0.503 0.117 C.170 C.972 C.298 0.042 0.574 C.C65 0.225 0.766 0.57C 0.520
 0.996 C.292 C.79C C.111 C.556 0.337 0.C12 0.C42 C.143 0.482 0.607 0.302
 0.353 C.397 0.206 C.662 0.119 0.754 C.450 C.918 C.453 0.463 0.699 0.022
 0.842 C.859 C.577 C.725 C.163 0.450 0.232 0.349 C.0C0 0.864 0.181 0.311
 C.236 0.622 0.607 0.C42 0.787 C.348 0.C06 C.9C4 C.365 0.053 0.037 0.745
 0.136 0.113 0.455 C.7C8 C.158 0.572 0.012 0.928 C.455 C.381 0.193 0.728
 0.629 C.220 0.657 0.962 C.86C C.501 C.268 C.C98 C.181 0.203 0.588 0.701
 0.909 0.148 0.7C8 C.9C9 0.C88 0.345 0.277 0.556 C.840 0.033 0.639 0.539
 0.482 0.041 C.907 0.C77 C.299 0.C97 0.892 C.478 C.835 0.7C7 0.733 0.029
 0.581 C.224 C.112 0.659 C.945 0.741 0.94C C.969 C.360 0.434 0.365 0.285
 0.422 C.967 0.005 0.328 0.924 0.595 C.253 C.157 0.668 0.594 0.554 0.984
 0.913 0.622 0.516 C.502 0.364 C.667 0.724 C.344 0.546 C.178 0.151 0.302
 0.457 C.021 0.019 C.923 0.365 0.882 C.01C C.121 0.637 0.734 0.671 0.416
 0.459 0.CC9 C.919 0.434 C.331 0.C79 0.50C C.284 C.209 0.694 0.283 0.454
 0.178 0.978 0.272 0.827 0.512 0.634 0.195 C.462 C.019 0.956 0.560 0.761
 0.524 C.294 0.C47 0.634 0.382 0.591 0.103 C.3C3 C.889 0.607 0.638 0.367
 0.463 C.471 0.664 0.742 C.476 C.178 0.785 C.113 0.610 0.646 0.390 0.520
 C.611 C.988 0.431 C.699 0.312 0.580 0.672 0.810 C.814 C.597 0.256 C.164
 0.679 0.6C3 0.504 C.595 0.C33 0.846 0.783 C.C79 C.430 C.868 0.343 0.244
 0.376 0.C62 C.99C C.381 C.371 0.801 0.467 C.592 C.348 0.759 0.422 0.697
 0.388 0.055 0.836 C.518 C.585 C.842 0.793 C.177 0.926 0.964 0.450 0.022
 0.C85 0.311 0.102 C.816 C.973 0.494 0.208 0.8C3 C.948 0.462 0.242 0.287
 0.546 0.698 0.269 0.339 0.607 0.594 0.102 C.266 0.677 0.668 0.913 C.462
 0.562 C.2C7 0.188 C.264 C.895 C.991 0.893 C.442 C.615 C.7C9 0.722 C.950
 0.208 C.696 0.304 0.557 0.605 0.617 0.256 C.984 0.595 0.715 0.936 0.178
 0.141 C.153 0.654 0.544 C.376 C.363 0.793 C.492 C.812 0.447 0.376 0.231
 0.644 0.263 0.785 C.341 C.982 C.829999999
----
UNIFORM-GALSS TEST  211
-0.283 C.916 0.776 C.69C C.910 0.506 C.816 C.348 C.659-0.301 0.630-0.397
 0.515 C.227 1.253 0.421 0.499 0.288 1.189-0.264 0.099 0.051 0.411 0.951
 0.273 0.154 0.861 C.937 C.446 0.702 1.451 C.C35 C.515 C.770 0.559 1.053
-1.157 0.9C2 0.533 1.270 0.761 1.110 1.19C 0.433 C.573 C.374 1.317 1.255
 0.547 1.145 0.667-0.077 C.422-0.159-0.037 C.C88 0.406 0.849 0.898 0.372
-0.324 0.025 0.632 0.365 0.375 0.694-0.206 0.126-0.381 1.149 0.983 1.184
-C.011 C.653 0.266 1.C35 C.536 C.936 1.177 1.644 0.782 0.198 0.222 0.445
 0.714 C.6C7 0.374 C.341 C.79C 0.302 1.075 C.2C4 0.436 C.887 0.234 0.874
 0.048 0.938 0.733-0.340-0.C12 0.497 0.418 C.840 C.091 0.578 0.606-0.340
 0.656 C.66C 0.584 C.837 C.454 C.695 0.606 0.053-0.276 1.600 1.394-0.038
 1.173 0.762 0.642 0.185-C.C23 0.037 0.508 C.313-0.718-C.249 0.124-0.096
-0.299 0.196 1.086 C.487 0.317 0.635 0.462 C.559 C.181 1.799 0.287 0.583
 0.313 0.387 1.C67 C.996 C.7C2 C.068-0.227 C.198 0.305-0.021 0.849 1.063
 1.049 0.226-0.297 C.930 0.828 C.884 1.217-0.391 0.007-0.004 1.238 0.376
 0.418 1.075 0.C83-0.020 0.362 0.601 0.037 0.634 0.109 0.524 1.356 1.024
-0.602 C.763 1.261 0.3C2-0.C63 0.704 0.446-C.410 0.401 0.7C4 1.070-0.023
-0.064 1.C87-0.737-0.476 1.156 0.648 0.624 0.257 0.643 0.147 C.719 0.174
-0.552-C.183 0.583 C.740 C.592-0.144 0.222 0.388 0.563 0.933 1.108 1.022
 0.818 C.686 0.683 0.514 C.284-C.280 0.358 1.2C3-0.643 0.110 0.012 0.399
 C.899 C.4CC 0.994 0.880 C.743 0.1C2 1.12C C.391 0.191 0.196 1.176 0.149
 0.512 1.132 C.916 C.838 C.445 1.330 0.563 C.610 C.699 0.675 0.310 C.586
 0.448-C.475 0.317 C.858 C.839-0.297 0.214 C.985 C.484 1.004 1.598 0.494
 0.467 1.188 0.536 C.381 1.339-0.011 0.C64 C.113 C.619 0.604 0.687 0.622
 C.297 C.2C3 0.378 1.313 C.829 0.422 0.C78 C.C57 0.143 0.868-0.302 0.693
 0.633 1.116 0.118-C.469 C.663 0.7C8 0.685 C.850 0.566 0.657 1.217 0.394
 0.643-C.055-C.C00 0.881 1.163 0.520 0.787 1.493 1.366 C.801 0.301 1.384
-0.541 1.176 0.236 0.675 1.119 1.C00 0.250 0.457-C.01C 0.098 0.975 0.288
 0.686 C.764 0.007 0.697 C.789 0.259 0.414 C.68C 0.852 0.315 0.231 0.203
 1.394 C.131 0.963 C.699 C.404-C.124 0.583 0.071 1.838-0.313-0.467 0.191
 0.125 1.67C 0.224 0.4C0 0.658 0.900 1.034 0.C05 0.801 C.920-0.188 0.786
 C.387 1.243 C.875 0.989 0.718-0.152 0.C09 1.329 0.562 0.687 0.968 0.490
 0.16C C.177-C.025 1.125 C.217 1.206 1.221-C.145-C.088 0.629-0.131 0.272
 0.388 0.772 1.C46-C.067 C.76C 0.428 0.852 C.867 C.610 0.359 1.352 0.571
-0.011-0.2C5 1.084 C.CC9 0.917 0.438 1.C50 0.616 C.884 0.954-0.275 1.295
 0.346-1.018 1.C49 0.417 1.23C 1.127 1.435 C.823-0.069 0.464 1.261-0.179
 0.812 C.537 0.875 C.190 0.707 0.857 0.C94 0.176 C.916 0.406 0.204 0.494
 1.709 0.133 0.460 C.828-0.174 0.457 0.584 C.832-C.026 1.265-0.232-0.373
-0.172 C.913 0.673 C.303 0.035 1.226-0.072 C.920 1.207-0.003 0.632 0.004
 0.951 C.3C7 0.798 1.479 0.196 1.058 0.873 C.060 C.524 0.501 0.373 0.954
-0.C72 0.988 0.351 0.C53 0.248 C.430-0.379 C.533 0.688 0.524 C.447 1.376
-0.263-0.064 C.301 1.486 C.351 0.806-0.374 0.735 0.958-0.262 0.332 0.963
999999
```

Figure 33.

Output

Description

The output from the program KOLM gives the statistics and probability statements described below, and in addition identifies the distribution functions being considered. Sorted samples are printed on option.

The following items are produced as output:

1. Z score, where

   $z = \sqrt{n}\, D_n$ for the one-sample test; n is the sample size, and D_n is the maximum difference between the empirical distribution function and the theoretical distribution function.

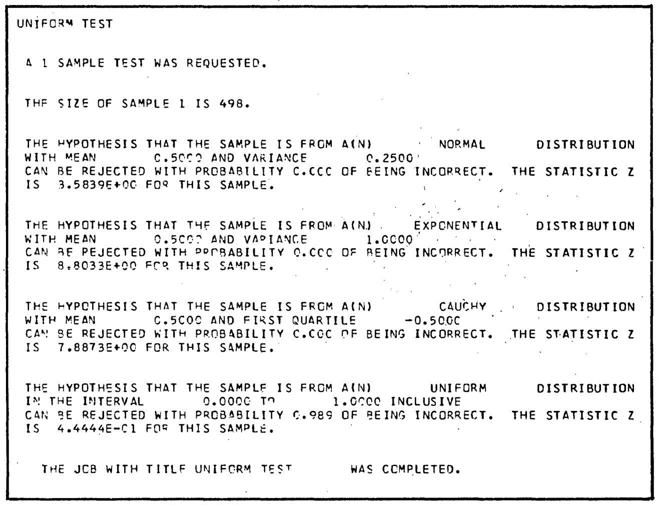
   $$z = \sqrt{\frac{mn}{m+n}}\; D_{m,n} \text{ for the two-sample test;}$$

   m is the size of the second sample; n is the size of the first sample; $D_{m,n}$ is the maximum difference between the two empirical distribution functions.

2. The probability of incorrectly rejecting the hypothesis of equality of distribution functions.

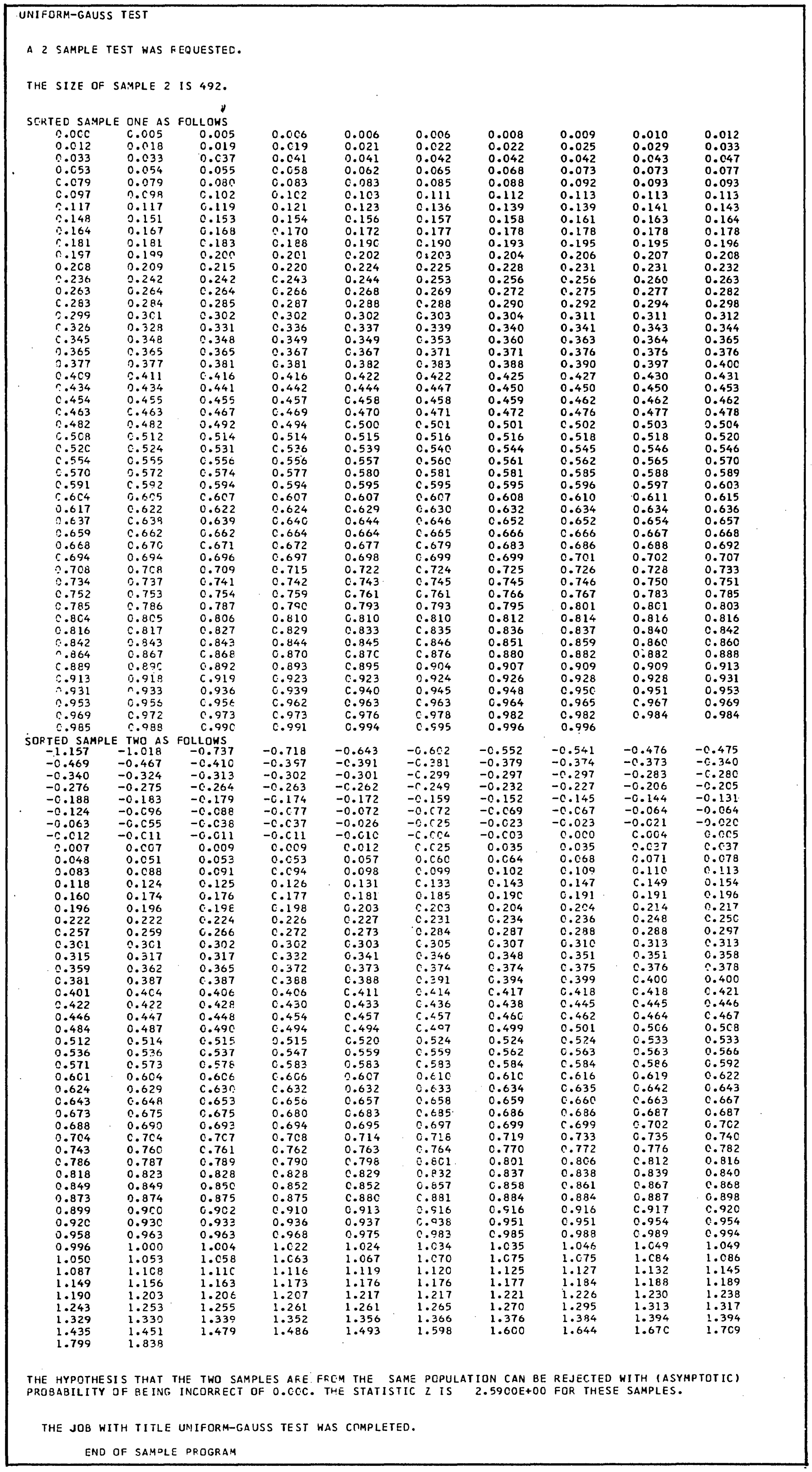
Sample

Sample output is shown in Figure 34.



```
UNIFORM TEST


 A 1 SAMPLE TEST WAS REQUESTED.


 THE SIZE OF SAMPLE 1 IS 498.


 THE HYPOTHESIS THAT THE SAMPLE IS FROM A(N)        NORMAL       DISTRIBUTION
 WITH MEAN         C.5CCC AND VARIANCE        C.2500
 CAN BE REJECTED WITH PROBABILITY C.CCC OF BEING INCORRECT.  THE STATISTIC Z
 IS  3.5839E+0C FOR THIS SAMPLE.


 THE HYPOTHESIS THAT THE SAMPLE IS FROM A(N)     EXPONENTIAL     DISTRIBUTION
 WITH MEAN         0.5CCC AND VARIANCE        1.CC00
 CAN BE REJECTED WITH PROBABILITY 0.CCC OF BEING INCORRECT.  THE STATISTIC Z
 IS  8.8033E+00 FOR THIS SAMPLE.


 THE HYPOTHESIS THAT THE SAMPLE IS FROM A(N)        CAUCHY       DISTRIBUTION
 WITH MEAN         C.5C0C AND FIRST QUARTILE      -0.500C
 CAN BE REJECTED WITH PROBABILITY C.CCC OF BEING INCORRECT.  THE STATISTIC Z
 IS  7.8873E+0C FOR THIS SAMPLE.


 THE HYPOTHESIS THAT THE SAMPLE IS FROM A(N)       UNIFORM       DISTRIBUTION
 IN THE INTERVAL         0.000C TO        1.0CCC INCLUSIVE
 CAN BE REJECTED WITH PROBABILITY C.989 OF BEING INCORRECT.  THE STATISTIC Z
 IS  4.4444E-C1 FOR THIS SAMPLE.


   THE JCB WITH TITLE UNIFCRM TEST        WAS CCMPLETED.
```

Figure 34.



```
UNIFORM-GAUSS TEST

 A 2 SAMPLE TEST WAS REQUESTED.

 THE SIZE OF SAMPLE 2 IS 492.

 SORTED SAMPLE ONE AS FOLLOWS
       0.000    0.005    0.005    0.006    0.006    0.006    0.008    0.009    0.010    0.012
       0.012    0.018    0.019    0.019    0.021    0.022    0.022    0.025    0.029    0.033
       0.033    0.033    0.037    0.041    0.041    0.042    0.042    0.042    0.043    0.047
       0.053    0.054    0.055    0.058    0.062    0.065    0.068    0.073    0.073    0.077
       0.079    0.079    0.080    0.083    0.083    0.085    0.088    0.092    0.093    0.093
       0.097    0.098    0.102    0.102    0.103    0.111    0.112    0.113    0.113    0.113
       0.117    0.117    0.119    0.121    0.123    0.136    0.139    0.139    0.141    0.143
       0.148    0.151    0.153    0.154    0.156    0.157    0.158    0.161    0.163    0.164
       0.164    0.167    0.168    0.170    0.172    0.177    0.178    0.178    0.178    0.178
       0.181    0.181    0.183    0.188    0.190    0.190    0.193    0.195    0.195    0.196
       0.197    0.199    0.200    0.201    0.202    0.203    0.204    0.206    0.207    0.208
       0.208    0.209    0.215    0.220    0.224    0.225    0.228    0.231    0.231    0.232
       0.236    0.242    0.242    0.243    0.244    0.253    0.256    0.256    0.260    0.263
       0.263    0.264    0.264    0.266    0.268    0.269    0.272    0.275    0.277    0.282
       0.283    0.284    0.285    0.287    0.288    0.288    0.290    0.292    0.294    0.298
       0.299    0.301    0.302    0.302    0.302    0.303    0.304    0.311    0.311    0.312
       0.326    0.328    0.331    0.336    0.337    0.339    0.340    0.341    0.343    0.344
       0.345    0.348    0.348    0.349    0.349    0.353    0.360    0.363    0.364    0.365
       0.365    0.365    0.365    0.367    0.367    0.371    0.371    0.376    0.376    0.376
       0.377    0.377    0.381    0.381    0.382    0.383    0.388    0.390    0.397    0.400
       0.409    0.411    0.416    0.416    0.422    0.422    0.425    0.427    0.430    0.431
       0.434    0.434    0.441    0.442    0.444    0.447    0.450    0.450    0.450    0.453
       0.454    0.455    0.455    0.457    0.458    0.458    0.459    0.462    0.462    0.462
       0.463    0.463    0.467    0.469    0.470    0.471    0.472    0.476    0.477    0.478
       0.482    0.482    0.492    0.494    0.500    0.501    0.501    0.502    0.503    0.504
       0.508    0.512    0.514    0.514    0.515    0.516    0.516    0.518    0.518    0.520
       0.520    0.524    0.531    0.536    0.539    0.540    0.544    0.545    0.546    0.546
       0.554    0.555    0.556    0.556    0.557    0.560    0.561    0.562    0.565    0.570
       0.570    0.572    0.574    0.577    0.580    0.581    0.581    0.585    0.588    0.589
       0.591    0.592    0.594    0.594    0.595    0.595    0.595    0.596    0.597    0.603
       0.604    0.605    0.607    0.607    0.607    0.607    0.608    0.610    0.611    0.615
       0.617    0.622    0.622    0.624    0.629    0.630    0.632    0.634    0.634    0.636
       0.637    0.638    0.639    0.640    0.644    0.646    0.652    0.652    0.654    0.657
       0.659    0.662    0.662    0.664    0.664    0.665    0.666    0.666    0.667    0.668
       0.668    0.670    0.671    0.672    0.677    0.679    0.683    0.686    0.688    0.692
       0.694    0.694    0.696    0.697    0.698    0.699    0.699    0.701    0.702    0.707
       0.708    0.708    0.709    0.715    0.722    0.724    0.725    0.726    0.728    0.733
       0.734    0.737    0.741    0.742    0.743    0.745    0.745    0.746    0.750    0.751
       0.752    0.753    0.754    0.759    0.761    0.761    0.766    0.767    0.783    0.785
       0.785    0.786    0.787    0.790    0.793    0.793    0.795    0.801    0.801    0.803
       0.804    0.805    0.806    0.810    0.810    0.810    0.812    0.814    0.816    0.816
       0.816    0.817    0.827    0.829    0.833    0.835    0.836    0.837    0.840    0.842
       0.842    0.843    0.843    0.844    0.845    0.846    0.851    0.859    0.860    0.860
       0.864    0.867    0.868    0.870    0.870    0.876    0.880    0.882    0.882    0.888
       0.889    0.890    0.892    0.893    0.895    0.904    0.907    0.909    0.909    0.913
       0.913    0.918    0.919    0.923    0.923    0.924    0.926    0.928    0.928    0.931
       0.931    0.933    0.936    0.939    0.940    0.945    0.948    0.950    0.951    0.953
       0.953    0.956    0.956    0.962    0.963    0.963    0.964    0.965    0.967    0.969
       0.969    0.972    0.973    0.973    0.976    0.978    0.982    0.982    0.984    0.984
       0.985    0.988    0.990    0.991    0.994    0.995    0.996    0.996
 SORTED SAMPLE TWO AS FOLLOWS
      -1.157   -1.018   -0.737   -0.718   -0.643   -0.602   -0.552   -0.541   -0.476   -0.475
      -0.469   -0.467   -0.410   -0.397   -0.391   -0.381   -0.379   -0.374   -0.373   -0.340
      -0.340   -0.324   -0.313   -0.302   -0.301   -0.299   -0.297   -0.297   -0.283   -0.280
      -0.276   -0.275   -0.264   -0.263   -0.262   -0.249   -0.232   -0.227   -0.206   -0.205
      -0.188   -0.183   -0.179   -0.174   -0.172   -0.159   -0.152   -0.145   -0.144   -0.131
      -0.124   -0.096   -0.088   -0.077   -0.072   -0.072   -0.069   -0.067   -0.064   -0.064
      -0.063   -0.055   -0.038   -0.037   -0.026   -0.025   -0.023   -0.023   -0.021   -0.020
      -0.012   -0.011   -0.011   -0.011   -0.010   -0.004   -0.003    0.000    0.004    0.005
       0.007    0.007    0.009    0.009    0.012    0.025    0.035    0.035    0.037    0.037
       0.048    0.051    0.053    0.053    0.057    0.060    0.064    0.068    0.071    0.078
       0.083    0.088    0.091    0.094    0.098    0.099    0.102    0.109    0.110    0.113
       0.118    0.124    0.125    0.126    0.131    0.133    0.143    0.147    0.149    0.154
       0.160    0.174    0.176    0.177    0.181    0.185    0.190    0.191    0.191    0.196
       0.196    0.196    0.198    0.198    0.203    0.203    0.204    0.204    0.214    0.217
       0.222    0.222    0.224    0.226    0.227    0.231    0.234    0.236    0.248    0.250
       0.257    0.259    0.266    0.272    0.273    0.284    0.287    0.288    0.288    0.297
       0.301    0.301    0.302    0.302    0.303    0.305    0.307    0.310    0.313    0.313
       0.315    0.317    0.317    0.332    0.341    0.346    0.348    0.351    0.351    0.358
       0.359    0.362    0.365    0.372    0.373    0.374    0.374    0.375    0.376    0.378
       0.381    0.387    0.387    0.388    0.388    0.391    0.394    0.399    0.400    0.400
       0.401    0.404    0.406    0.406    0.411    0.414    0.417    0.418    0.418    0.421
       0.422    0.422    0.428    0.430    0.433    0.436    0.438    0.445    0.445    0.446
       0.446    0.447    0.448    0.454    0.457    0.457    0.460    0.462    0.464    0.467
       0.484    0.487    0.490    0.494    0.494    0.497    0.499    0.501    0.506    0.508
       0.512    0.514    0.515    0.515    0.520    0.524    0.524    0.524    0.533    0.533
       0.536    0.536    0.537    0.547    0.559    0.559    0.562    0.563    0.563    0.566
       0.571    0.573    0.578    0.583    0.583    0.583    0.584    0.584    0.586    0.592
       0.601    0.604    0.606    0.606    0.607    0.610    0.610    0.616    0.619    0.622
       0.624    0.629    0.630    0.632    0.632    0.633    0.634    0.635    0.642    0.643
       0.643    0.648    0.653    0.656    0.657    0.658    0.659    0.660    0.663    0.667
       0.673    0.675    0.675    0.680    0.683    0.685    0.686    0.686    0.687    0.687
       0.688    0.690    0.693    0.694    0.695    0.697    0.699    0.699    0.702    0.702
       0.704    0.704    0.707    0.708    0.714    0.718    0.719    0.733    0.735    0.740
       0.743    0.760    0.761    0.762    0.763    0.764    0.770    0.772    0.776    0.782
       0.786    0.787    0.789    0.790    0.798    0.801    0.801    0.806    0.812    0.816
       0.818    0.823    0.828    0.828    0.829    0.832    0.837    0.838    0.839    0.840
       0.849    0.849    0.850    0.852    0.852    0.857    0.858    0.861    0.867    0.868
       0.873    0.874    0.875    0.875    0.880    0.881    0.884    0.884    0.887    0.898
       0.899    0.900    0.902    0.910    0.913    0.916    0.916    0.916    0.917    0.920
       0.920    0.930    0.933    0.936    0.937    0.938    0.951    0.951    0.954    0.954
       0.958    0.963    0.963    0.968    0.975    0.983    0.985    0.988    0.989    0.994
       0.996    1.000    1.004    1.022    1.024    1.034    1.035    1.046    1.049    1.049
       1.050    1.053    1.058    1.063    1.067    1.070    1.075    1.075    1.084    1.086
       1.087    1.108    1.110    1.116    1.119    1.120    1.125    1.127    1.132    1.145
       1.149    1.156    1.163    1.173    1.176    1.176    1.177    1.184    1.188    1.189
       1.190    1.203    1.206    1.207    1.217    1.217    1.221    1.226    1.230    1.238
       1.243    1.253    1.255    1.261    1.261    1.265    1.270    1.295    1.313    1.317
       1.329    1.330    1.339    1.352    1.356    1.366    1.376    1.384    1.394    1.394
       1.435    1.451    1.479    1.486    1.493    1.598    1.600    1.644    1.670    1.709
       1.799    1.838

 THE HYPOTHESIS THAT THE TWO SAMPLES ARE FROM THE  SAME POPULATION CAN BE REJECTED WITH (ASYMPTOTIC)
 PROBABILITY OF BEING INCORRECT OF 0.000. THE STATISTIC Z IS   2.5900E+00 FOR THESE SAMPLES.

  THE JOB WITH TITLE UNIFORM-GAUSS TEST WAS COMPLETED.

        END OF SAMPLE PROGRAM
```

Figure 34. (Continued)

Program Modifications

1. Program capacity may be increased or decreased by making changes in the allocation statements. If this is done, the limits on the DO statements may require modification, as will be the case if data formats require changing. It is also possible that output formats may require changes.
2. Any modifications to the subroutine KLMO in terms of added continuous pdf's should be reflected in the program KOLM. It may be necessary to:
   a. Modify the declaration of DIST (5, 3), which contains the switches calling on pdf's and also contains the parameters u and s used by KLMO.
   b. Modify the pdf titling cards numbered KOLM 230 through 270.
   c. Modify the section of the program from S70 through S100 to reflect changes a and b above. These statements call KLMO to perform tests and output results.
3. List of variables in KOLM, and their usage:

D -	Temporary vector used in reading samples
DAS2 -	Job Control Card name (----)
DIST -	5 by 3 matrix. The five elements in column 1 are switches that allow the 5 pdf's to be used in one-sample tests. Columns 2 and 3 contain the parameters u and s for the associated test.
ERROR -	Error (in using KLMO, used for skipping the test concerned)
I -	Counter used to print correct pdf name for pdf used in the test
IFL -	Error indicator (job deck error)
IES -	Error (in using KLMO, used for error message)
IO -	Switch (used for printing samples)
IR -	Number of samples to be read in current job
IS -	Number of samples to be used in current job (1 or 2)
M -	Size of the second sample
N -	Size of first sample
P -	Probability of being incorrect if hypothesis is rejected
S2 -	Temporary storage for u and s output
TIT1 -	Current pdf names
TITLE -	Job title
X -	Sample 1
Y -	Sample 2
Z -	Z statistic from KLMO or KLM2

Operating Instructions

This sample program is a standard PL/I program and needs no special operating instructions. Data set SYSIN is used for input; data set SYSPRINT, for output.

Error Messages

The following error conditions will result in messages, followed by the action specified:

1. Neither a one-nor two-sample test is requested, or the size of either sample is larger than 500 -- CC. 21, CONTROL CARD, INCORRECT, OR SAMPLE SIZE TOO LARGE. JOB IGNORED. Action: Cards are read until a new job control card is found, or until the hopper is empty.
2. Sample size is less than 100 (not an error) -- NOTE THE REMARKS CONCERNING ASYMPTOTIC RESULTS AND SAMPLE SIZE IN SUBROUTINE SMIR. Action: none. Job continues.
3. The requirement of subroutine KLMO that certain parameters be nonzero or positive is violated -- AT LEAST ONE (S) ENTRY PARAMETER FOR THE SUBROUTINE KLMO WAS INCORRECT. THE TEST FOR THE ASSOCIATED CONTINUOUS PDF WAS IGNORED. Action: All tests are made for cases where the parameter s was correct.
4. A case in which an error requires aborting the job, and the succeeding job in the job stack requests a two-sample test where the second sample is to be compared with a (first) sample, which was read on the previous job -- THIS JOB CALLS FOR THE USE OF A PREVIOUSLY READ SAMPLE, AND THE PREVIOUS JOB WAS IGNORED BECAUSE OF ERRORS. JOB IGNORED. Action: Cards are read until a new job control card is found, or until the hopper is empty.
5. The job control card preceding a job is not there or is incorrect -- FIRST CARD IN JOB DECK (JOB CONTROL CARD) IS INCORRECT. Action: Cards are read until a new job control card is found, or until the hopper is empty.

Timing

The execution time of this program on a System /360 Model 40, using a 2540 Card Reader as input and a 1403 Printer, Model N1, as output, is 35 seconds for job 1 and 55 seconds for job 2.

```
KOLM..
/**********************************************************************/
/*                                                                    */
/* THE PURPOSE OF THIS ROUTINE IS TO:                                 */
/*        (1) READ THE CONTROL CARD FOR A ONE OR TWO SAMPLE TEST.     */
/*        (2) READ THE SAMPLE DATA AND DETERMINE THE SAMPLE SIZES.    */
/*        (3) CALL SMIR, KLMO, AND KLM2 FOR CALCULATION OF            */
/*            PROBABILITIES.                                          */
/*        (4) PRINT RESULTS.                                          */
/*                                                                    */
/**********************************************************************/
   PROCEDURE OPTIONS (MAIN),.
   DECLARE
        (DASH,DAS2) CHARACTER (4),
        (I,J,K,L,M,N,IS,IR,IO,IFL,E) FIXED BINARY,
        (DIST(5,3),D(12),X(501),Y(501),P,Z,S2) FLOAT BINARY,
        TITLE CHARACTER (20),
        TIT1(5) CHARACTER (16),
        IES CHARACTER (1),
        ERROR EXTERNAL CHARACTER (1),.
   ON ENDFILE(SYSIN) GO TO S200,.
   SW   =0,.
   IFL  =0,.
   DASH ='----',.                          /* INITIALIZE NAMES           */
   TIT1(1)   ='     NORMAL     ',.         /* AND JOB CONTROL CARD       */
   TIT1(2)   ='  EXPONENTIAL   ',.
   TIT1(3)   ='     CAUCHY     ',.
   TIT1(4)   ='     UNIFORM    ',.
   TIT1(5)   =' USER-SPECIFIED ',.
S10..
   GET EDIT(DAS2,E)(A(4),X(75),F(1)),.
   IF DASH=DAS2
   THEN                                    /* READ TITLE AND             */
S20..                                      /* PROGRAM PARAMETERS         */
        DO,.
        GET EDIT(TITLE,IS,IR,IO,DIST(*,*),E)(A(20),3 F(1),5(F(1),2 F(5)
            ),X(1),F(1)),.
        IES  ='0',.
        PUT EDIT (TITLE)(A(20)) PAGE,.
        PUT EDIT (' A',IS,' SAMPLE TEST WAS REQUESTED.')(SKIP(3),A(2),
            F(2),A(27)),.
        IF SW=0 AND IS=2 AND IR=1
        THEN DO,.
             PUT EDIT(' FIRST JOB REQUIRES PREVIOUS DATA FOR A TWO SAM'
                 ,'PLE TEST.')(SKIP(3),A(47),A(9)),.
             SW   =1,.
             GO TO S40,.
             END,.
        SW   =1,.
        IF IR=0                            /* NO. OF SAMPLES DECISION    */
        THEN IF IS GE 1
             THEN GO TO S140,.
             ELSE                          /* NO. OF SAMPLES WRONG       */
S30..
                  DO,.
                  PUT EDIT(' CC.21 OF THE PROGRAM CONTROL CARD IS INCO'
                      ,'RRECT.  JOB IGNORED.')(SKIP(3),A(42),A(20)),.
S40..
                  GET EDIT(DAS2,E)(A(4),X(75),F(1)),.
                  IF DASH=DAS2
                  THEN DO,.
                       IFL  =1,.
                       GO TO S20,.
                       END,.
                  ELSE GO TO S40,.
                  END,.
        ELSE IF IFL NE 0
             THEN DO,.                     /* ERROR IN PREVIOUS JOB      */
                  PUT EDIT(' THIS JOB CALLS FOR THE USE OF A PREVIOUSL'
                      ,'Y READ SAMPLE, AND THE PREVIOUS JOB WAS IGNORE'
                      ,'D BECAUSE OF ERRORS.','JOB IGNORED.')(SKIP(3),
                      A(42),A(46),A(20),SKIP,A(13)),.
                  GO TO S40,.
                  END,.
             ELSE GO TO S180,.
        END,.
   ELSE DO,.
        PUT EDIT(' FIRST CARD IN JOB DECK (JOB CONTROL CARD) IS INCORR'
            ,'ECT.')(SKIP(3),A(52),A(4)),.
        GO TO S40,.
        END,.
S50..
   IF IS=2
   THEN GO TO S180,.
   ELSE IF IS GT 2
        THEN GO TO S30,.
        ELSE GO TO S65,.
S60..
   IF IS LE 1                              /* ONE SAMPLE TEST USING ALL  */
   THEN DO,.                               /* DISTRIBUTIONS REQUESTED    */
S65..
        DO I=1 TO 5,.
        IF DIST(I,1) NE 0
        THEN GO TO S70,.
        END,.
        PUT EDIT(' NO PDF COMPARISON IS ASKED FOR.')(SKIP(3),A(32)),.
S70..
        DO I=1 TO 5 ,.
        IF DIST(I,1) = 1
        THEN CALL KLMO(X,N,Z,P,I,DIST(I,2),DIST(I,3)),.
        IF ERROR='0' OR ERROR='3'
        THEN
             DO,.                          /* OUTPUT RESULTS             */
             PUT EDIT(' THE HYPOTHESIS THAT THE SAMPLE IS FROM A(N)    '
                 ,TIT1(I),' DISTRIBUTION' )(SKIP(3),A(47),A(16),A(13))
                 ,.
            IF I LT 3                      /* PREPARE OUTPUT             */
            THEN DO,.
                 S2   =DIST(I,3)**2,.
                 PUT EDIT(' WITH MEAN',DIST(I,2),' AND VARIANCE',S2)
                     (SKIP,A(10),F(13,4),A(13),F(13,4)),.
                 GO TO S80,.
                 END,.
            ELSE IF I=3
                 THEN DO,.
                      S2   =DIST(I,2)-DIST(I,3),.
                      PUT EDIT(' WITH MEAN',DIST(I,2),' AND FIRST ',
                          'QUARTILE',S2)(SKIP,A(10),F(13,4),A(11),
                          A(8),F(13,4)),.
                      GO TO S80,.
                      END,.
                 ELSE IF I LE 4
                      THEN DO,.
                           PUT EDIT(' IN THE INTERVAL',DIST(I,2),' TO'
                               ,DIST(I,3),' INCLUSIVE')(SKIP,A(16),
                               F(13,4),A(3),F(13,4),A(10)),.
S80..
                           PUT EDIT(' CAN BE REJECTED WITH PROBABILIT'
                               ,'Y',P,' OF BEING INCORRECT.  THE STAT'
                               ,'ISTIC Z',' IS',Z,' FOR THIS SAMPLE.')
                               (SKIP,A(32),A(1),F(6,3),A(30),A(7),SKIP
                               ,A(3),E(12,4),A(17)),.
                           END,.
                      GO TO S90,.
             END,.
        ELSE IES  =ERROR,.
S90..
        END,.
        END,.
   ELSE GO TO S110,.
   IF IO NE 0                              /* OUTPUT SAMPLE ONE DECISION */
   THEN DO,.
        PUT EDIT (' SORTED SAMPLE ONE FOLLOWS')(SKIP(3),A(26)),.
        PUT EDIT ((X(J) DO J=1 TO N))(SKIP,10 (F(10,3))),.
        END,.
   IF IES    ='0'
   THEN
S100..
        DO,.
        IFL  =0,.
        PUT EDIT ('   THE JOB WITH TITLE',TITLE,' WAS COMPLETED.')
            (SKIP(3),A(22),A(18),A(15)),.
        IF ERROR='3'
        THEN PUT EDIT ('NOTE THE REMARKS CONCERNING ASYMPTOTIC RESULTS'
             ,' AND SAMPLE SIZE IN SUBROUTINE SMIR.')(SKIP(3),A(46),
             A(36)),.
        GO TO S10,.
        END,.
   ELSE DO,.
        PUT EDIT(' AT LEAST ONE (S) ENTRY PARAMETER FOR THE SUBROUTINE'
            ,' KLMO WAS INCORRECT. ',' TEST FOR THE ASSOCIATED CONTINU'
            ,'OUS PDF WAS IGNORED.')(SKIP(3),A(52),A(21),SKIP,A(32),
            A(20)),.
        GO TO S100,.
        END,.
S110..
   CALL KLM2(X,Y,N,M,Z,P),.                /* TWO SAMPLE TEST            */
   IF IO=0                                 /* OUTPUT SAMPLES DECISION    */
   THEN
S120..
        DO,.
        PUT EDIT(' THE HYPOTHESIS THAT THE TWO SAMPLES ARE FROM THE ',
            ' SAME POPULATION CAN BE REJECTED WITH (ASYMPTOTIC)',
            ' PROBABILITY OF BEING INCORRECT OF',P,'. THE STATISTIC Z '
            ,'IS ',Z,' FOR THESE SAMPLES.')(SKIP(3),A(50),A(50),SKIP,
            A(34),F(6,3),A(18),A(3),E(12,4),A(19)),.
        GO TO S100,.
        END,.
   ELSE
S130..
        DO,.
        PUT EDIT (' SORTED SAMPLE ONE AS FOLLOWS')(SKIP(3),A(29)),.
        PUT EDIT ((X(J) DO J=1 TO N))(SKIP,10 F(10,3)),.
        PUT EDIT (' SORTED SAMPLE TWO AS FOLLOWS')(SKIP(3),A(29)),.
        PUT EDIT ((Y(J) DO J=1 TO M))(SKIP,10 F(10,3)),.
        GO TO S120,.
        END,.
S140..
   N    =0,.                               /* READ FIRST SAMPLE          */
        DO I=1 TO 50,.
        GET EDIT((D(K) DO K=1 TO 12),E)(12 F(6),X(7),F(1)),.
             DO J=1 TO 12,.
             IF D(J) = 999999.0            /* CHECK FOR END OF SAMPLE    */
             THEN GO TO S170,.
             N    =N+1,.
             IF N GE 501                   /* MAXIMUM SAMPLE SIZE        */
             THEN
S150..
                  DO,.
                  PUT EDIT (' SAMPLE SIZE IS TOO LARGE.  JOB IGNORED.')
                      (SKIP(3),A(43)),.
                  GO TO S40,.
                  END,.
             X(N) =D(J),.                  /* PLACE SAMPLE IN X          */
             END,.
        END,.
S170..
   PUT EDIT(' THE SIZE OF SAMPLE 1 IS',N,'.')(SKIP(3),A(24),F(4),A(1))
        ,.
   GO TO S50,.
S180..
   M    =0,.                               /* READ SECOND SAMPLE         */
        DO I=1 TO 50,.
        GET EDIT((D(K) DO K=1 TO 12),E)(12 F(6),X(7),F(1)),.
             DO J=1 TO 12,.
             IF D(J)=999999.0              /* CHECK FOR END OF SAMPLE    */
             THEN GO TO S190,.
             M    =M+1,.
             IF M GE 501                   /* MAX SIZE OF SAMPLE         */
             THEN DO,.
                  PUT EDIT(' SAMPLE SIZE IS TOO LARGE.  JOB IGNORED.')
                      (SKIP(3),A(43)),.
                      GO TO S40,.
                      END,.
             Y(M) =D(J),.                  /* PLACE SAMPLE IN Y          */
             END,.
        END,.
S190..
   PUT EDIT(' THE SIZE OF SAMPLE 2 IS',M,'.')(SKIP(3),A(24),F(4),A(1))
        ,.
   GO TO S60,.
S200..
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')
        (SKIP(2),COLUMN(10),A),.
   END,.                                   /* END OF PROCEDURE KOLM      */
```

TRIPLE EXPONENTIAL SMOOTHING EXPN

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 the main program, named EXPN, a special input routine, named DAT3, and one subroutine from the Scientific Subroutine Package: EXSM.

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

> (12F(6,0)) format for input data cards. Therefore, if a problem satisfies the above conditions, the sample program need not be modified. However, if input data cards are prepared using a different format, the input format in the input routine DAT3 must be modified. The general rules for program modification are described later.

Input

Control Card

One control card is required for each problem and is read by the main program, EXPN. 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, α $(0.0 < \alpha < 1.0)$	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

Smoothing constant and three coefficients must be keypunched with decimal points.

Leading zeros do not have to be keypunched.

Data Cards

Time series data are keypunched using the format (12 F(6,0)). This format assumes that each data point is keypunched in a six-column field, with twelve fields per card.

Data Setup

The deck setup is shown in Figure 35.

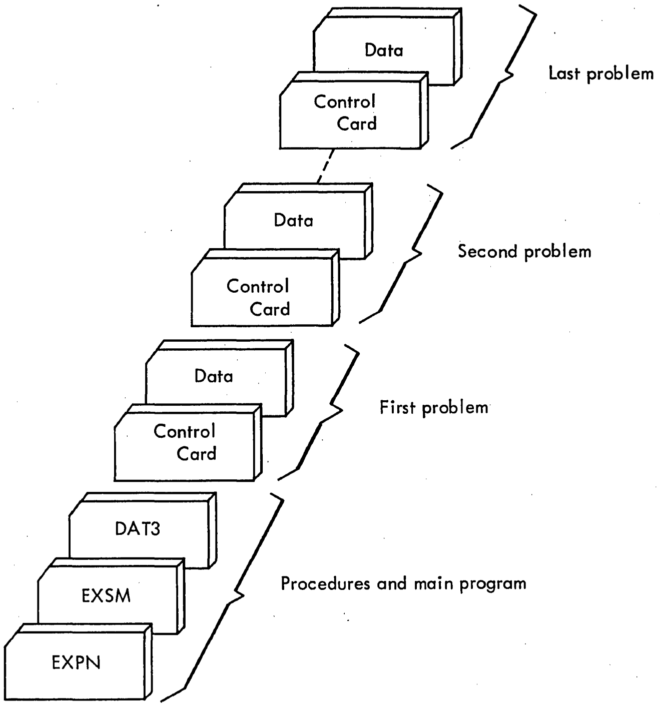
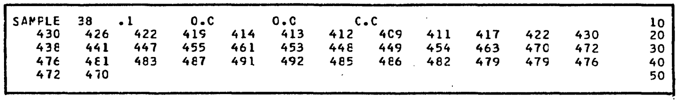


Figure 35.

Sample

The listing of input cards for the sample problem is shown in Figure 36.



```
SAMPLE  38   .1        0.C        0.C        C.C                                       10
   430   426   422   419   414   413   412   4C9   411   417   422   430              20
   438   441   447   455   461   453   448   449   454   463   47C   472              30
   476   481   483   487   491   492   485   486   482   479   479   476              40
   472   470                                                                          50
```

Figure 36.

Output

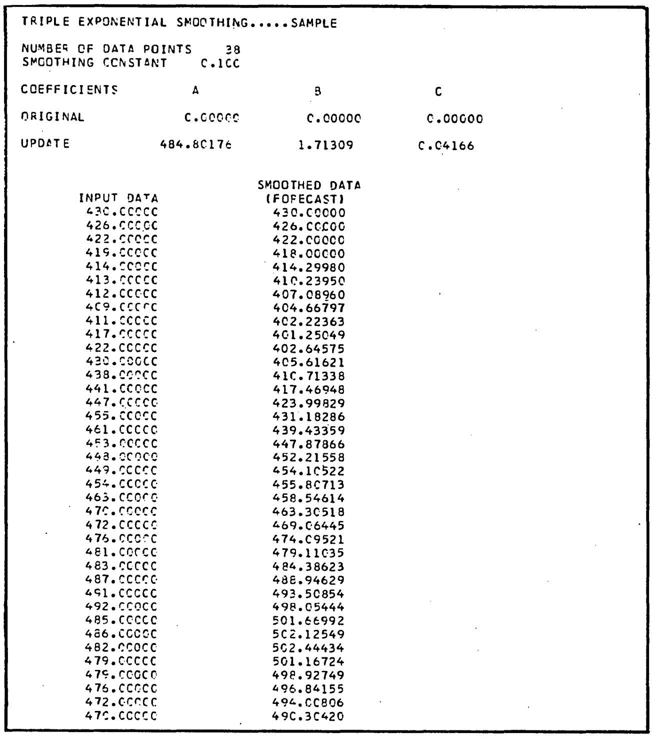
Description

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 37.



```
TRIPLE EXPONENTIAL SMOOTHING.....SAMPLE

NUMBER OF DATA POINTS    38
SMOOTHING CCNSTANT    C.1CC

COEFFICIENTS          A               B              C

ORIGINAL          C.CCCCC         C.00000        C.00000

UPDATE          484.8C176         1.71309        C.C4166


                         SMOOTHED DATA
       INPUT DATA         (FORECAST)
        43C.CCCCC          430.C0000
        426.CCCCC          426.CCC00
        422.CCCCC          422.C0000
        419.CCCCC          418.00C00
        414.CCCCC          414.29980
        413.CCCCC          41C.23950
        412.CCCCC          407.08960
        4C9.CCCCC          404.66797
        411.CCCCC          4C2.22363
        417.CCCCC          4C1.25049
        422.CCCCC          402.64575
        43C.COOCC          4C5.61621
        438.CCCCC          41C.71338
        441.CCOCC          417.46948
        447.CCCCC          423.99829
        455.CCOCC          431.18286
        461.CCCCC          439.43359
        453.CCCCC          447.87866
        448.CCCCC          452.21558
        449.CCCCC          454.1C522
        454.CCCCC          455.8C713
        463.CCOCC          458.54614
        47C.CCCCC          463.3C518
        472.CCCCC          469.C6445
        476.CCOCC          474.C9521
        481.COCCC          479.11C35
        483.CCCCC          484.38623
        487.CCCCC          488.94629
        491.CCCCC          493.5C854
        492.CCOCC          498.C5444
        485.CCCCC          501.66992
        486.CCCCC          5C2.12549
        482.CCOCC          5C2.44434
        479.CCCCC          501.16724
        479.CCOCO          498.92749
        476.CCCCC          496.84155
        472.CCCCC          494.CC806
        47C.CCCCC          49C.3C420
```

Figure 37.

Program Modifications

Input data in a different format can also be handled by providing a different 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 input format statement of the input routine DAT3.

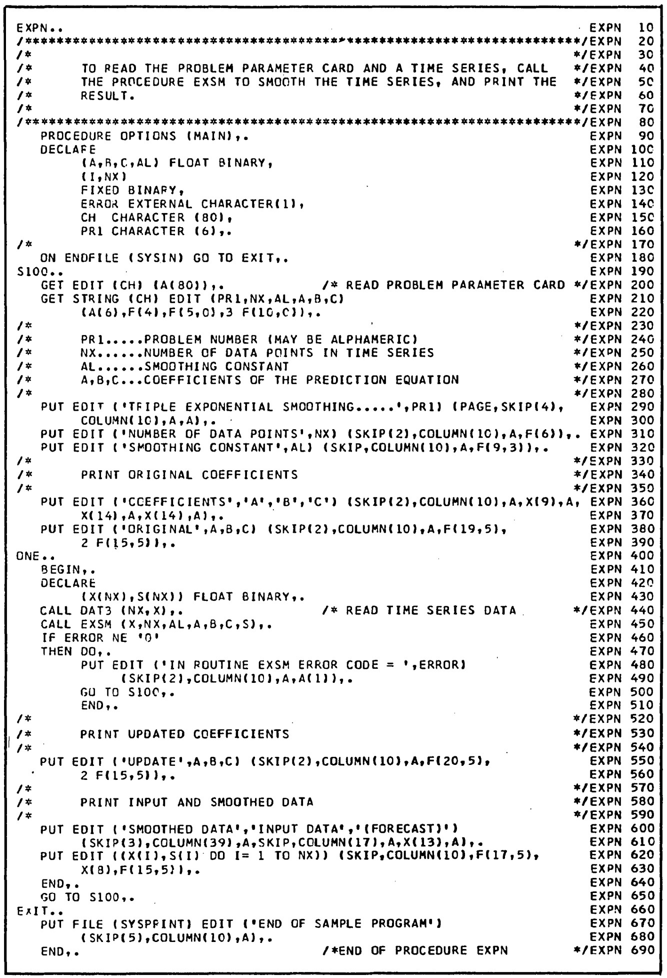
Only the format statement and the variables per card count indicator (NF), which appears in subroutine DAT3, can be changed. Since sample data are three-digit numbers, rather than using six-column fields, as in the sample problem, each data point might have been keypunched in a three-column field, with 24 fields per card. If so, the format is changed to (24 F(3,0)) and the variables per card count indicator (NF) is changed to agree with the number of variables per data card.

Operating Instructions

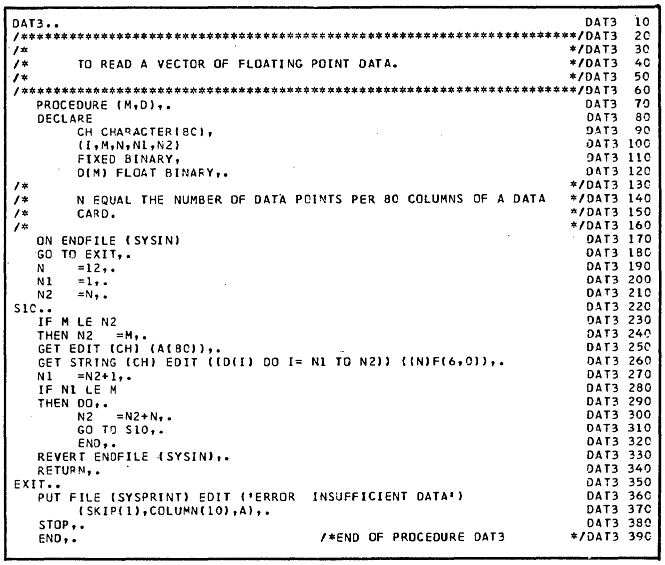
The sample program for triple exponential smoothing is a standard PL/I program. Special operating instructions are not required. Data set SYSIN is used for input; data set SYSPRINT, for output.

Timing

The execution of this sample program on a System/360 Model 40, using a 2540 Card Reader as input and a 1403 Printer, Model N1, as output, is 33 seconds.



```
EXPN..
/**********************************************************************/
/*                                                                    */
/*      TO READ THE PROBLEM PARAMETER CARD AND A TIME SERIES, CALL    */
/*      THE PROCEDURE EXSM TO SMOOTH THE TIME SERIES, AND PRINT THE   */
/*      RESULT.                                                       */
/*                                                                    */
/**********************************************************************/
   PROCEDURE OPTIONS (MAIN),.
   DECLARE
        (A,B,C,AL) FLOAT BINARY,
        (I,NX)
        FIXED BINARY,
        ERROR EXTERNAL CHARACTER(1),
        CH  CHARACTER (80),
        PR1 CHARACTER (6),.
/*                                                                    */
   ON ENDFILE (SYSIN) GO TO EXIT,.
S100..
   GET EDIT (CH) (A(80)),.              /* READ PROBLEM PARAMETER CARD */
   GET STRING (CH) EDIT (PR1,NX,AL,A,B,C)
        (A(6),F(4),F(5,0),3 F(10,C)),.
/*                                                                    */
/*      PR1.....PROBLEM NUMBER (MAY BE ALPHAMERIC)                    */
/*      NX......NUMBER OF DATA POINTS IN TIME SERIES                  */
/*      AL......SMOOTHING CONSTANT                                    */
/*      A,B,C...COEFFICIENTS OF THE PREDICTION EQUATION               */
/*                                                                    */
   PUT EDIT ('TRIPLE EXPONENTIAL SMOOTHING.....',PR1) (PAGE,SKIP(4),
        COLUMN(10),A,A),.
   PUT EDIT ('NUMBER OF DATA POINTS',NX) (SKIP(2),COLUMN(10),A,F(6)),.
   PUT EDIT ('SMOOTHING CONSTANT',AL) (SKIP,COLUMN(10),A,F(9,3)),.
/*                                                                    */
/*      PRINT ORIGINAL COEFFICIENTS                                   */
/*                                                                    */
   PUT EDIT ('CCEFFICIENTS','A','B','C') (SKIP(2),COLUMN(10),A,X(9),A,
        X(14),A,X(14),A),.
   PUT EDIT ('ORIGINAL',A,B,C) (SKIP(2),COLUMN(10),A,F(19,5),
        2 F(15,5)),.
ONE..
   BEGIN,.
   DECLARE
        (X(NX),S(NX)) FLOAT BINARY,.
   CALL DAT3 (NX,X),.                   /* READ TIME SERIES DATA       */
   CALL EXSM (X,NX,AL,A,B,C,S),.
   IF ERROR NE '0'
   THEN DO,.
        PUT EDIT ('IN ROUTINE EXSM ERROR CODE = ',ERROR)
             (SKIP(2),COLUMN(10),A,A(1)),.
        GO TO S100,.
        END,.
/*                                                                    */
/*      PRINT UPDATED COEFFICIENTS                                    */
/*                                                                    */
   PUT EDIT ('UPDATE',A,B,C) (SKIP(2),COLUMN(10),A,F(20,5),
        2 F(15,5)),.
/*                                                                    */
/*      PRINT INPUT AND SMOOTHED DATA                                 */
/*                                                                    */
   PUT EDIT ('SMOOTHED DATA','INPUT DATA','(FORECAST)')
        (SKIP(3),COLUMN(39),A,SKIP,COLUMN(17),A,X(13),A),.
   PUT EDIT ((X(I),S(I) DO I= 1 TO NX)) (SKIP,COLUMN(10),F(17,5),
        X(8),F(15,5)),.
   END,.
   GO TO S100,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('END OF SAMPLE PROGRAM')
        (SKIP(5),COLUMN(10),A),.
   END,.                                /*END OF PROCEDURE EXPN        */
```



```
DAT3..
/**********************************************************************/
/*                                                                    */
/*      TO READ A VECTOR OF FLOATING POINT DATA.                      */
/*                                                                    */
/**********************************************************************/
   PROCEDURE (M,D),.
   DECLARE
        CH CHARACTER(8C),
        (I,M,N,N1,N2)
        FIXED BINARY,
        D(M) FLOAT BINARY,.
/*                                                                    */
/*      N EQUAL THE NUMBER OF DATA POINTS PER 8C COLUMNS OF A DATA    */
/*      CARD.                                                         */
/*                                                                    */
   ON ENDFILE (SYSIN)
   GO TO EXIT,.
   N    =12,.
   N1   =1,.
   N2   =N,.
S1C..
   IF M LE N2
   THEN N2   =M,.
   GET EDIT (CH) (A(8C)),.
   GET STRING (CH) EDIT ((D(I) DO I= N1 TO N2)) ((N)F(6,0)),.
   N1   =N2+1,.
   IF N1 LE M
   THEN DO,.
        N2   =N2+N,.
        GO TO S10,.
        END,.
   REVERT ENDFILE (SYSIN),.
   RETURN,.
EXIT..
   PUT FILE (SYSPRINT) EDIT ('ERROR  INSUFFICIENT DATA')
        (SKIP(1),COLUMN(10),A),.
   STOP,.
   END,.                               /*END OF PROCEDURE DAT3        */
```

ALLOCATION OF OVERHEAD COSTS (COST)

Problem Description

A standard problem in finance is the allocation of overhead costs (for example, electricity, transportation, . . .) to productive (charge) departments.

Overhead costs are initially charged to auxiliary departments. The costs of the auxiliary departments must be distributed among the productive departments using a given allocation key. For any auxiliary department the allocation key gives the distribution of unit costs among all departments (productive and auxiliary).

The problem is to calculate a transition matrix that can be used to obtain the final cost allocation to productive departments (by multiplying it with the given cost vector).

Mathematical Background

The calculation procedure is best described using matrix notation.

Let n be the number of auxiliary departments and m the number of productive departments.

The given allocation keys form a matrix K of dimension n+m by n, where the i-th column gives the distribution of unit costs of the i-th auxiliary department among all m+n departments.

$$\begin{array}{c} \text{n-columns} \\ K = \begin{pmatrix} R \\ S \end{pmatrix} \begin{matrix} \} \text{ n-rows} \\ \} \text{ m-rows} \end{matrix} \end{array} \qquad (1)$$

K (given) is segmented into two parts, R and S.

R is of dimension n by n and S of dimension m by n. R contains the allocation keys for charging auxiliary departments by an auxiliary department, while S contains the allocation keys for charging productive departments by an auxiliary department.

If R is null, S is already the required transition matrix.

Note that all elements of K are nonnegative and that the sum of all elements in any column is one.

Let C be the vector of dimension n+m containing the costs of auxiliary departments (first n elements) and the costs of productive departments (last m elements):

$$C = \begin{pmatrix} CA \\ CP \end{pmatrix} \begin{matrix} \} \; n \\ \} \; m \end{matrix} \qquad (2)$$

Distributing overhead costs CA according to allocation key K gives a new vector

$$C_1 = \begin{pmatrix} CA_1 \\ CP_1 \end{pmatrix} \text{ with } \begin{matrix} CA_1 = R \cdot CA \\ CP_1 = S \cdot CA + CP \end{matrix} \qquad (3)$$

and by iteration

$$C_k = \begin{pmatrix} CA_k \\ CP_k \end{pmatrix} \text{ with } \begin{matrix} CA_k = R \circ CA_{k-1} = R^k \circ CA \\ CP_k = S \cdot CA_{k-1} + CP_{k-1} \end{matrix} \qquad (3)$$

A realistic allocation key requires each auxiliary department to allot part of its costs to productive departments.

Under this assumption for the elements r_{ik} of R.

$$0 \le r_{ik} \le \alpha < 1 \text{ for all } \begin{matrix} i = 1, 2, \ldots, n \\ k = 1, 2, \ldots, n \end{matrix} \qquad (4)$$

and

$$\sum_{i=1}^{n} r_{ik} \le \alpha < 1 \text{ for all } k = 1, 2, \ldots, n \qquad (5)$$

This means $R^k \to 0$ for $k \to \infty$ and I-R is nonsingular.

Therefore, iteration (3) will give the allocation of costs C to productive departments.

One step is sufficient if R = 0 (when no auxiliary department is charging an auxiliary department again).

The process (3) is easily described in matrix notation:

$$C_0 = C, \quad C_k = \begin{pmatrix} R & 0 \\ S & I \end{pmatrix} \cdot C_{k-1} = \begin{pmatrix} R & 0 \\ S & I \end{pmatrix}^k \cdot C_0 \qquad (6)$$

Therefore:

$$\lim_{k \to \infty} \begin{pmatrix} R & 0 \\ S & I \end{pmatrix}^k =$$

$$\lim_{k \to \infty} \begin{pmatrix} R^k & 0 \\ S(I+R+\ldots+R^{k-1}) & I \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ T & I \end{pmatrix}$$

defines the desired transition matrix T, which will give the final cost allocation with a single matrix multiplication:

$$T = \lim_{k \to \infty} S \cdot (I+R+\ldots+R^{k-1}) = S \cdot (I-R)^{-1}$$

The rows of T may be calculated one at a time from

$$T^T = (I-R)^{-T} \cdot S^T \qquad (7)$$

Programming Considerations

Calculation of T is done in two major steps:

1. The matrix $(I-R)^T$ is factored into a product of two triangular matrices $L \cdot U = (I-R)^T$.
2. The column vectors of S^T (that is, row vectors of S) are divided by the triangular factors L and U.

Doing the second step sequentially, one column at a time, saves considerable storage space, since the only data needed in core simultaneously is an n by n matrix, containing $(I-R)^T$, the triangular factors L and U, and a vector of dimension n. This allows calculation of the transition matrix T, which allocates overhead costs, for a very large number of charge departments, as long as the number of auxiliary departments is of moderate size.

Program

The program for allocation of overhead costs consists of the main program, COST, and two procedures from the Scientific Subroutine Package:

MFG -- triangular factorization of a general matrix

MDLG-- division by triangular factors from left-hand side

Capacity

The limitation on the number of auxiliary departments depends on the size of storage available for data. The number of productive departments is not limited by core size.

Dynamic storage allocation is used for data arrays with extent n+1 by n.

Input

One control card is required for each data set. This card is prepared as follows:

Columns	Contents	For Sample Problem
1-10	Problem number (may be alphameric)	HILBERT
11-15	Number of auxiliary departments	6
16-20	Number of productive departments	4

Leading zeros do not have to be keypunched.

Data Cards

The rows of matrix $K = \begin{pmatrix} R \\ S \end{pmatrix}$ are read into the computer one at a time.

The elements are keypunched in successive cards, assuming six 10-column fields per card. These fields are 11-20, 21-30, 31-40, 41-50, 51-60, 61-70. Columns 1-4 are used for identification of the row. Each row must start with a new card. An input format of F(10, 8) is used for the ten-column fields.

Deck setup is shown in Figure 38.

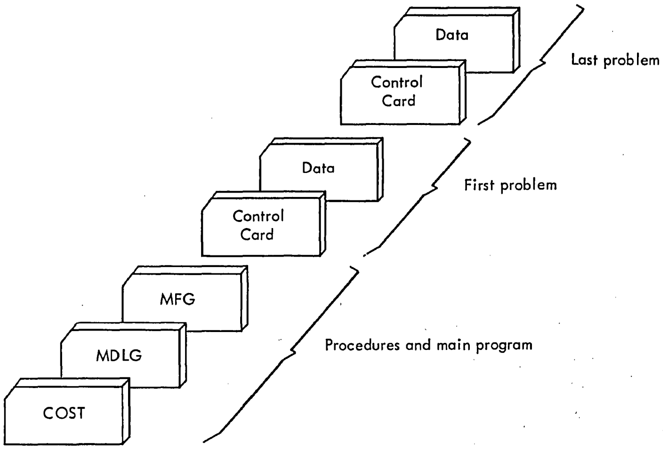
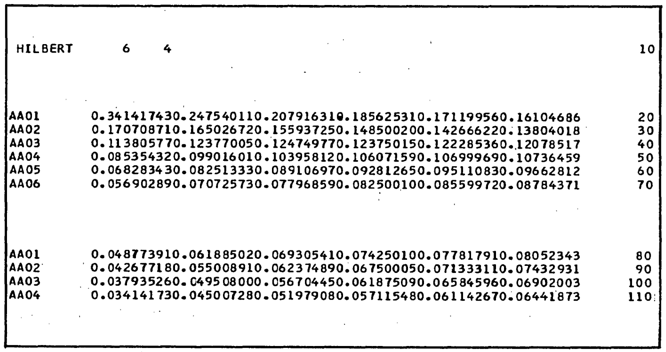


Figure 38.

Sample

A listing of input cards for the sample problem is shown in Figure 39.



```
HILBERT      6    4                                                          10


AA01      0.341417430.247540110.207916310.185625310.171199560.16104686      20
AA02      0.170708710.165026720.155937250.148500200.142666220.13804018      30
AA03      0.113805770.123770050.124749770.123750150.122285360.12078517      40
AA04      0.085354320.099016010.103958120.106071590.106999690.10736459      50
AA05      0.068283430.082513330.089106970.092812650.095110830.09662812      60
AA06      0.056902890.070725730.077968590.082500100.085599720.08784371      70


AA01      0.048773910.061885020.069305410.074250100.077817910.08052343      80
AA02      0.042677180.055008910.062374890.067500050.071333110.07432931      90
AA03      0.037935260.049508000.056704450.061875090.065845960.06902003     100
AA04      0.034141730.045007280.051979080.057115480.061142670.06441873     110
```

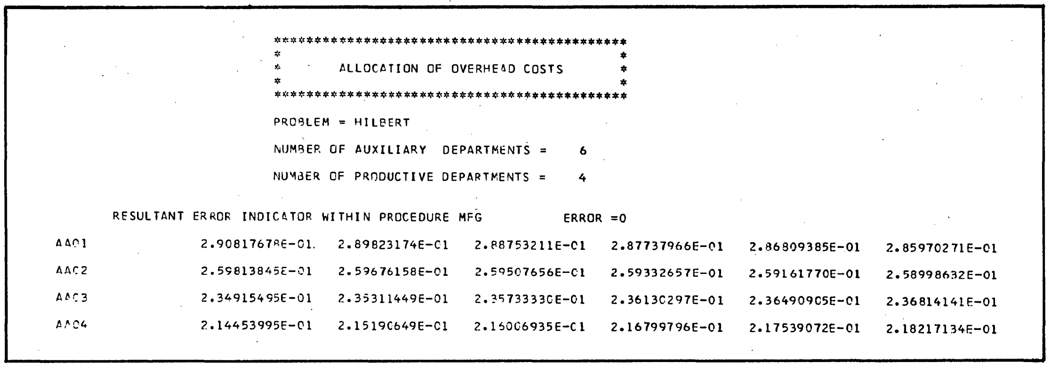
Figure 39.

Output

As output, the resulting transition matrix T is listed rowwise.

Sample

The output listing for the sample problem is shown in Figure 40.



```
                    *******************************************
                    *                                         *
                    *       ALLOCATION OF OVERHEAD COSTS      *
                    *                                         *
                    *******************************************

                    PROBLEM = HILBERT

                    NUMBER OF AUXILIARY  DEPARTMENTS =    6

                    NUMBER OF PRODUCTIVE DEPARTMENTS =    4

        RESULTANT ERROR INDICATOR WITHIN PROCEDURE MFG          ERROR =0

AA01             2.90817678E-01.  2.89823174E-01   2.88753211E-01   2.87737966E-01   2.86809385E-01   2.85970271E-01

AA02             2.59813845E-01   2.59676158E-01   2.59507656E-01   2.59332657E-01   2.59161770E-01   2.58998632E-01

AA03             2.34915495E-01   2.35311449E-01   2.35733330E-01   2.36130297E-01   2.36490905E-01   2.36814141E-01

AA04             2.14453995E-01   2.15190649E-01   2.16006935E-01   2.16799796E-01   2.17539072E-01   2.18217134E-01
```

Figure 40.

Program Modifications

Input data in a different format can be handled by providing different formats in corresponding GET EDIT statements.

Error Messages

The value of the error indicator as set by procedure MFG is included in the listing:

ERROR = 'O' means successful factorization.
ERROR = 'P' means incorrect value N.
ERROR = 'S' means incorrect data matrix R. (I-R) is singular.
ERROR = 'C' means (I-R) is nearly singular. To avoid a breakdown of the method, input data has been slightly modified.
ERROR = 'W' means (I-R) is nearly singular. Results may have poor accuracy.
In the case ERROR = 'S', calculation is bypassed.

Operating Instructions

The sample program for overhead cost allocation is a standard PL/I procedure. Special operating instructions are not required. Data set SYSIN is used for input; and data set SYSPRINT, for output.

Timing

The execution time of this sample program on a System/360 Model 40, using a 2540 Card Reader and a 1403 Printer, Model N2, as output, is 19 seconds.

```
COST..                                                                  COST  10
/**********************************************************************/COST  20
/*                                                                    */COST  30
/*                      ALLOCATION OF OVERHEAD COSTS                  */COST  40
/*                                                                    */COST  50
/**********************************************************************/COST  60
   PROCEDURE OPTIONS(MAIN),.                                            COST  70
   DECLARE                                                              COST  80
        ERROR EXTERNAL CHARACTER(1),  /*EXTERNAL ERROR INDICATOR      */COST  90
        (CNR,CHNF) CHARACTER(10),                                       COST 100
        CH CHARACTER(1),                                                COST 110
        EPS BINARY FLOAT,                                               COST 120
        (I,IND,K,L,M,N)                                                 COST 130
        BINARY FIXED,.                                                  COST 140
   ON ENDFILE (SYSIN) GO TO BACK,.                                      COST 150
START..                               /*SET EPS FOR INTERNAL TEST FOR*/COST 160
   EPS  =1E-6,.                       /*LOSS OF SIGNIFICANT DIGITS   */COST 170
   GET EDIT                                                             COST 180
        (CNR,N,M,CH)                  /*READ NUMBER OF COLUMNS, ROWS */COST 190
        (A(10),F(5),F(5),X(59),A(1)),.                                  COST 200
   PUT EDIT                           /*WRITE HEADING                */COST 210
        ('*******************************************',                 COST 220
        '*                                         *',                  COST 230
        '*       ALLOCATION OF OVERHEAD COSTS      *',                  COST 240
        '*                                         *',                  COST 250
        '*******************************************')                  COST 260
        (PAGE,SKIP(2),(5)(X(30),A,SKIP)),.                              COST 270
   PUT EDIT                                                             COST 280
        ('PROBLEM =',CNR,'NUMBER OF AUXILIARY  DEPARTMENTS =',N,        COST 290
        'NUMBEP OF PRODUCTIVE DEPAPTMENTS =',M)                         COST 300
        (SKIP(2),X(30),A,A,(2)(SKIP(2),X(30),A,F(5))),.                 COST 310
        BEGIN,.                                                         COST 320
        DECLARE                                                         COST 330
             (R(N,N),S(N,1),                                            COST 340
             W(N) DEFINED S(1SUB,1))                                    COST 350
             BINARY FLOAT,            /*SINGLE PRECISION VERSION  /*S*/COST 360
 /*          BINARY FLOAT(53),        /*DOUBLE PRECISION VERSION  /*D*/COST 370
             IPER(N)                                                    COST 380
             BINARY FIXED,.                                             COST 390
        IND  =1,.                     /*CALCULATE VALUES FOR INPUT   */COST 400
        L    =N,.                     /*FORMAT LIST                  */COST 410
             DO WHILE (L GT 6),.                                        COST 420
             L    =L-6,.                                                COST 430
             IND  =IND+1,.            /*IND MEANS THE NUMBER OF CARDS*/COST 440
             END,.                    /*FOR ONE ROW OF R             */COST 450
        L    =(6-L)*10,.              /*L SPECIFIES HORIZONT. SPACING*/COST 460
             DO I =1 TO N,.           /*EXECUTE LOOP OVER ROWS OF R  */COST 470
             GET EDIT                 /*READ I-TH ROW OF MATRIX R    */COST 480
                  (CHNR,W)                                              COST 490
                  (A(10),(IND)((6)F(10,8),X(20))),.                     COST 500
             GET EDIT                 /*HORIZONTAL SPACING           */COST 510
                  (CNR)                                                 COST 520
                  (X(L),A(10)),.                                        COST 530
             W(I) =W(I)-1,.           /*COMPUTE  TRANSPOSED (U-R)    */COST 540
             R(*,I)=-W,.              /*WHERE U MEANS UNIT MATRIX    */COST 550
             END,.                                                      COST 560
        CALL MFG(R,IPER,N,EPS),.      /*CALL FACTORIZATION PROCEDURE */COST 570
        PUT EDIT                      /*WRITE ERROR INDICATOR OF MFG */COST 580
             ('RESULTANT ERROR INDICATOR WITHIN PROCEDURE MFG',         COST 590
             'ERFOR =',ERROP)(SKIP(3),X(10),A,X(10),A,A),.              COST 600
             DO I =1 TO M,.           /*EXECUTE LOOP OVER ROWS OF S  */COST 610
             GET EDIT                 /*READ ANY ROW OF MATRIX S     */COST 620
                  (CHNR,W)                                              COST 630
                  (A(10),(IND)((6)F(10,8),X(20))),.                     COST 640
             GET EDIT                                                   COST 650
                  (CNR)                                                 COST 660
                  (X(L),A(10)),.                                        COST 670
             IF ERROR NE 'S'                                            COST 680
             THEN DO,.                /*PERFORM MATRIX DIVISION      */COST 690
                  CALL MDLG(R,S,IPER,N,18,'0'),.                        COST 700
                  PUT EDIT            /*WRITE ALLOCATION ROW         */COST 710
                       (CHNR,W)                                         COST 720
                       (SKIP(2),X(3),A,X(5),(IND)((6)E(17,8),X(18))),. COST 730
                  END,.                                                 COST 740
             END,.                                                      COST 750
        END,.                                                           COST 760
   GO TO START,.                                                        COST 770
BACK..                                                                  COST 780
   END,.                              /*END OF PROCEDURE COST        */COST 790
```

GH20-0586-0

Printed in U.S.A. GH20-0586-0

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)