Aston University

Some pages of this thesis may have been removed for copyright restrictions.

If you have discovered material in Aston Research Explorer which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our Takedown policy and contact the service immediately (openaccess@aston.ac.uk)

MATHEMATICAL SOFTWARE FOR MICROCOMPUTERS:

THESIS SUBMITTED FOR Ph.D DEGREE 1981 (APPENDICES)

EMMANUEL AMANO ONIBERE

ABSTRACT

The first appendix describes how the library is installed and used. The others are the documentations of the various routines implementing the selected algorithms. Each routine is described and in most cases an example is given to show how it is used.

Areas in mathematical computation for which routines are available are algebraic linear systems,nonlinear functions, polynomials,quadrature,ordinary differential equations, minimization, least squares approximation and special functions.

KEY WORDS: MICROCOMPUTER, NUMERICAL, SOFTWARE, PORTABLE, FORTRAN.

CONTENTS

		PAGE
APPENDIX A:	INSTALLATION OF THE LIBRARY AND GENERAL USAGE	A1 - A11
APPENDIX B:	ALGEBRAIC LINEAR SYSTEMS	B1 - B29
APPENDIX C:	ROOTS OF NON-LINEAR FUNCTIONS	C1 - C20
APPENDIX D:	SUBROUTINES FOR QUADRATURE	D1 - D24
APPENDIX E:	SUBROUTINES FOR ORDINARY DIFFERENTIAL EQUATIONS	E1 - E21
APPENDIX F:	MINIMIZATION AND LEAST SQUARES APPROXIMATION	F1 - F22
APPENDIX G:	SPECIAL FUNCTIONS AND MACHINE CONSTANTS	G1 - G33

APPENDIX A:

INSTALLATION OF THE LIBRARY AND GENERAL USAGE

1. Installation of the library

The first step is to determine machine dependent quantities. Some functions to determine these quantities have been provided so that you need not know what these quantities stand for before you can install the library. The quantities to be determined are

i) base of floating-point arithmetic

ii) the number of digits in the mantissa of a floating-point number

iii) the highest exponent

iv) the lowest exponent

v) maximum integer representable

vi) minimum integer representable

vii) the number of digits in the mantissa of a double length floating-point number if double precision arithmetic is available.

viii) machine relative error

ix) machine relative error for double precision arithmetic if such arithmetic is available.

The library can be installed in the following manner:

a) No changes in Program A1 if double precision arithmetic is available, otherwise remove lines 1, 18, 19 and 22 and change 8 in line 6 to 7. Remember that the output unit number used here is one. If the one of your computer is different, then replace 1 in lines 8, 14 and 19 with your machine output unit number.

b) No changes in FUNCTION IF1MQ if double precision arithmetic is available, otherwise remove lines 2, 10, 27 and 47 - 53.

c) FUNCTION RF2MQ should be left unchanged.

d) FUNCTION DF2MQ is <u>not</u> required if double precision arithmetic is not available.

e) After these modifications have been made, Program A1, IF1MQ, RF2MQ, (DF2MQ if double precision arithmetic is available) should be compiled, linked and executed.

The following were the results produced by Texas Tx990/4 microcomputers which has double precision arithmetic.

MACH(2)	= 16	(BASE-β)
MACH(3)	= 6	(NUMBER OF BASE-β DIGITS IN MANTISSA)
MACH(4)	= 75	(MAX EXPONENT)
MACH(5)	= -78	(MIN EXPONENT)
MACH(6)	= 32767	(MAX INTEGER REPRESENTABLE)
MACH(7)	= -32768	(MIN INTEGER REPRESENTABLE)
MACH(8)	= 14	(NUMBER OF BASE-B DIGITS IN MANTISSA OF DOUBLE LENGTH NUMBERS).
RF2MQ	= 0.9536743E-06	(MACHINE RELATIVE ERROR)
DF2MQ	= 0.222044604925031D-15	(MACHINE RELATIVE ERROR, DOUBLE LENGTH)

Note that statements in brackets are not printed out. If you know all these quantities, then there is no need to follow steps a to e.

f) It is assumed that you know the standard output unit number of your system. For Tx990/4, it is 1.

g) Using the results obtained in step e, FUNCTIONS IF1MQ, RF2MQ, DF2MQ (if double precision arithmetic is available) are then modified to become:

i) FUNCTION IF1MQ

MACH(1) should be assigned the standard output unit number of the system. Using the results obtained for Texas Tx990/4 microcomputer, in step e, FUNCTION IF1MQ becomes.

```
INTEGER FUNCTION IF1MQ (I)
DIMENSION MACH(8)
DATA MACH/1, 16, 6, 75, -78, 32767, -32768, 14/
IF1MQ = MACH (I)
RETURN
END
```

Note that MACH(1) is now assigned the standard output unit number of the system. This must be done.

ii) FUNCTION RF2MQ

Using the results obtained for Tx990/4 microcomputer RF2MQ becomes:

```
REAL FUNCTION RF2MQ(R)
RF2MQ  =  0.9536743 E-06
RETURN
END.
```

Note that the variable R in the parameter list has nothing to do with the computation. It is there for RF2MQ to satisfy the definition of a function in FORTRAN.

iii) <u>FUNCTION DF2MQ</u> (If double precision arithmetic is available)

Using the results obtained for Tx990/4, microcomputer, DF2MQ becomes:

```
DOUBLE PRECISION FUNCTION DF2MQ (D)
DF2MQ = 0.2220446049250 31D-15
RETURN
END.
```

Note that the variable D in the parameter list has nothing to do with the computation. It is there for DF2MQ to satisfy the definition of a function in FORTRAN.

h) Test whether the modification has been properly done by recompiling Program A1 (after the 2 in line 6 has been change to 1), IF1MQ, RF2MQ and DF2MQ (if double precision arithmetic is available), linking and executing the linked output. The print out will now display MACH(1), which should be the standard output unit number. The other results should correspond with those obtained in step e.

i) The compiled output of FUNCTIONS IF1MQ, RF2MQ DF2MQ (if double precision arithmetic is available), should now be copied into one file which can be given the name BASE. Also include in the BASE file the compiled output of RF8SF and DF8SF (if double precision arithmetic is available). See appendix G for the listing of RF8SF and DF8SF.

j) The other routines can now be typed in from the listings provided into diskettes or cassettes. Remember that the lines must not be numbered. The numbering in the listings should therefore be discarded. Single and double precision versions of FUNCTIONS are provided and if double precision arithmetic is available then the double precision versions of FUNCTIONS should also be included in the library. Each routine with the auxiliary subroutines listed with it (if any) should be compiled and their compiled output placed in one file. This should be done for all the routines. The auxiliary subroutines (if any) listed with a subroutine are not called by any other subroutine except the one they are listed with. For subroutines only the single precision versions are provided. It is preferable for a file containing the object code of a subroutine and its auxiliary (if any) to have the same name as the subroutine.

k) Before compilation, the single precision version of the subroutines can be converted to partial double precision or double precision by the use of subroutine IF1CO which is described below.

2. USAGE OF SUBROUTINE IF1CO

After typing in the listed routines (excluding the line numbers), the subroutine IF1CO can be used to convert these single precision subroutines to partial double precision or double precision. It is also possible to avoid the use of IF1CO if the following process is followed.

a) To convert to partial double precision, lines that start with C* or C., should be ommitted. C- should be replaced with two spaces while lines that start with C+ should be activated by

replacing each C+ with two spaces. Also the line above each line that starts with C+ should be removed. This steps should be taken when typing in the listings.

b) To convert to double precision, lines that start with C+ should be omitted while each C. and C- should be replaced with two spaces. Lines that start with C* should be activated by replacing each C* with two spaces and removing the line above it. These steps should be taken when typing in the listings.

However, mistakes can easily be made if the conversion to partial double precision or double precision is done this way. Also no master library is kept. To avoid these problems SUBROUTINE IF1CO is provided. In writing subroutine IF1CO, some assumptions have been made and as a result the subroutine should be used with care.

i) It is assumed that an integer storage unit contains two characters. This is generally true for microcomputers.

ii) It is assumed that the statement:

```
READ (INPUT, 200, END = L1) (ICH2(k),  k=1, 36) where L1
```

is a label, is allowed. Although such a statement is not allowed in standard FORTRAN, most FORTRAN compilers do allow it.

iii) It is also assumed that there are up to seventy two characters in a line.

Example A2 shows how SUBROUTINE IF1CO can be used. The file from which a subroutine(s) is to be converted is assigned a logic unit number. The file to which the converted subroutine(s) is to be written is also assigned a logic unit number. In example A2, the

input file is allocated the input unit number 20, while the output unit number 21 is allocated to the output file. IFLAG is given the value 1, if conversion to partial double precision is desired and 2 if conversion to double precision is wanted.

3. General Usage of library

The routines available in the library are described in great details starting from appendix B to G. After a user has selected the necessary routine(s), the routine(s) must then be linked to the user's compiled program. The routines in file BASE must always be linked to the user's compiled program if any routine in the library is called or referenced by a user's program. Since different microcomputers have different ways of linking together user's subroutine, it is perhaps sufficient to say that each routine should be treated the same way a user treats his own routine in a different file from his own main program. Always remember that IF1MQ, RF2MQ, RF8SF, (DF2MQ, DF8SF) are in the base file BASE.

Program A1

```
      DOUBLE PRECISION DP2MQ,T
      REAL R
C
C  THE INTEGER MACHINE CONSTANTS ARE NOW BEING COMPUTED
C
      DO 10 I=2,6
      J=IF1MQ(I)
      WRITE(1,100)I,J
10    CONTINUE
C
C  COMPUTE MACHINE RELATIVE ERROR
C
      R=RP2MQ(0.0)
      WRITE(1,200)R
C
C  COMPUTE MACHINE RELATIVE ERROR IN DOUBLE PRECISION
C
      T=DP2MQ(0.0D 00)
      WRITE(1,300)T
100   FORMAT(5X,5HMACH(,I1,2H)=,I6/)
200   FORMAT(5X,6HRP2MQ=,E14.7/)
300   FORMAT(5X,6HDP2MQ=,D23.16/)
      STOP
      END
```

FUNCTION IF1MQ

```
      INTEGER FUNCTION IF1MQ(I)
      DOUBLE PRECISION DA
C
C   I=2    BASE
C   I=3    NUMBER OF BASE-B DIGITS
C   I=4    MAX EXPONENT (10**EMAX)
C   I=5    MIN EXPONENT (10**EMIN)
C   I=6    MAX INTEGER REPRESENTABLE
C   I=7    MIN INTEGER REPRESENTABLE
C   I=8    NUMBER OF BASE-B DIGITS FOR DOUBLE LENGTH
C
      IF(I.NE.6) GO TO 50
      IF1MQ=32767
      RETURN
50    IF(I.NE.7) GO TO 60
      IF1MQ=-32768
      RETURN
60    A=0.20E 01
      B=0.200E 01
100   IF(((A+0.100E 01)-A) .NE. 0.10E 01) GO TO 200
      A=0.20E 01*A
      GO TO 100
200   IF(((A+B)-A) .NE. 0.0E 00) GO TO 300
      B=0.20E 01*B
      GO TO 200
300   BASE=(A+B)-A
      IF(I.EQ.8) GO TO 800
      K=I-1
      GO TO(400,500,600,700),K
400   IF1MQ=INT(BASE)
      RETURN
500   IDIT=0
      A=0.10E 01
550   IDIT=IDIT+1
      A=A*BASE
      IF(((A+0.10E 01)-A) .EQ. 0.10E 01) GO TO 550
      IF1MQ=IDIT
      RETURN
600   IF(BASE.EQ.0.20E 01) IF1MQ=38
      IF(BASE.EQ.0.10E 02) IF1MQ=63
      IF(BASE.EQ.0.16E 02) IF1MQ=75
      RETURN
700   IF(BASE.EQ.0.20E 01) IF1MQ=-38
      IF(BASE.EQ.0.10E 02) IF1MQ=-64
      IF(BASE.EQ.0.16E 02) IF1MQ=-78
      RETURN
  800 IDIT=0
      DA=0.10D 01
  900 IDIT=IDIT+1
      DA=DA*DBLE(BASE)
      IF(((DA+0.10D 01)-DA) .EQ. 0.10D 01) GO TO 900
      IF1MQ=IDIT
      RETURN
      END
```

FUNCTIONS RF2MQ AND DF2MQ

```
      REAL FUNCTION RF2MQ(T)
      PRE=0.390625E-02
   10 PRE=PRE*0.50000E 00
      C=0.10E 01+PRE
      IF(PRE.LT.0.100E-10) GO TO 20
      IF(C .GT. 0.100E 01) GO TO 10
      RF2MQ=0.20E 01*PRE
      RETURN
   20 RF2MQ=0.10E-05
      RETURN
      END
      DOUBLE PRECISION FUNCTION DF2MQ(T)
      DOUBLE PRECISION C,PRE
      PRE=0.390625D-02
   10 PRE=PRE*0.50000D 00
      C=0.10D 01+PRE
      IF(PRE.LT.0.100D-20) GO TO 20
      IF(C .GT. 0.10D 01) GO TO 10
      DF2MQ=0.20D 01*PRE
      RETURN
   20 DF2MQ=0.10D-14
      RETURN
      END
```

Example A2 and SUBROUTINE IF1CO

```
      INTEGER INPUT,IOUT,IFLAG
C
C  INPUT= INPUT UNIT NUMBER AND IT MUST BE ASSIGNED TO
C         THE INPUT FILE WHICH CONTAINS THE ROUTINE(S) TO
C         BE CONVERTED.
C  IOUT = OUTPUT UNIT NUMBER AND IT MUST BE ASSIGNED TO
C         THE OUTPUT FILE TO WHICH THE CONVERTED ROUTINE(S)
C         IS TO BE WRITTEN.
C  IFLAG= IFLAG=1, MEANS CONVERT TO PARTIAL DOUBLE PRECISION.
C         IFLAG=2, MEANS CONVERT TO DOUBLE PRECISION.
C
      INPUT=20
      IOUT=21
      IFLAG=1
      CALL IF1CO(INPUT,IOUT,IFLAG)
      STOP
      END
      SUBROUTINE IF1CO(INPUT,IOUT,IFLAG)
      DIMENSION ICH1(36),ICH2(36)
      DATA ICP,ICS,ICM,ICD,IS/2HC+,2HC*,2HC-,2HC.,2H  /
      IN=INPUT
      IT=IOUT
      READ(IN,200,END=55)(ICH1(K),K=1,36)
      IF(IFLAG.EQ.2) GO TO 60
C  CONVERSION TO PARTIAL DOUBLE PRECISION
      IF(ICH1(1).EQ.ICM) ICH1(1)=IS
5     READ(IN,200,END=50)(ICH2(K),K=1,36)
      IF(ICH2(1).EQ.ICS) GO TO 5
      IF(ICH2(1).EQ.ICD) GO TO 5
      IF(ICH2(1).EQ.ICM) ICH2(1)=IS
      IF(ICH2(1).NE.ICP) GO TO 10
      ICH2(1)=IS
      GO TO 30
10    WRITE(IT,200)(ICH1(K),K=1,36)
30    DO 40 K=1,36
40    ICH1(K)=ICH2(K)
      GO TO 5
50    WRITE(IT,200)(ICH1(K),K=1,36)
55    RETURN
C  CONVERSION TO DOUBLE PRECISION
60    READ(IN,200,END=110)(ICH2(K),K=1,36)
      IF(ICH2(1).EQ.ICP) GO TO 60
      IF(ICH2(1).EQ.ICD) ICH2(1)=IS
      IF(ICH2(1).EQ.ICM) ICH2(1)=IS
      IF(ICH2(1).NE.ICS) GO TO 80
      ICH2(1)=IS
      GO TO 90
80    WRITE(IT,200)(ICH1(K),K=1,36)
90    DO 100 K=1,36
100   ICH1(K)=ICH2(K)
      GO TO 60
110   WRITE(IT,200)(ICH1(K),K=1,36)
200   FORMAT(36A2)
      RETURN
      END
```

APPENDIX B

ALGEBRAIC LINEAR SYSTEMS

The subroutines in this appendix are used to:

i) solve a system of simultaneous linear equations

$$Ax = b$$

where A is a given nxn matrix, b a given n-vector and x the unknown n-vector

ii) obtain the inverse of a real non singular nxn matrix

iii) obtain the determinant of a real nxn matrix.

Table B summarizes the action of the subroutines.

SUBROUTINE NAME	PURPOSE
RF1LE	Decomposes an nxn matrix A into LU where L is a unit lower matrix and U an upper matrix. A is over written by LU.
RF2LE	Solves LUx = b
RF3LE	Refines the solution of LUx = b
RF4LE	Solves tridiagonal systems
RF1IN	Computes the inverse of a non singular nxn matrix after decomposition.
RF1DE	Determines the determinant of a square matrix after decomposition.

TABLE B: A summary of subroutines for algebraic linear systems.

1. SUBROUTINE RF1LE

a) Purpose

The subroutine RF1LE decomposes a real nxn matrix A into LU where L is a unit lower matrix and U an upper matrix. A is overwritten by LU.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

```
SUBROUTINE RF1LE (ID, N, A, SC, L, IFAIL)
```

c) Description

This subroutine decomposes an nxn real matrix using Crout's algorithm (see Palett and Wang (129)). Partial pivoting is used and each row is scaled before decomposition, by dividing it with the absolute value of the element with the highest magnitude in that row. Such scaling helps in testing for near singularity.

d) Parameters

ID integer variable, which on entry, the user must set to the row size of matrix A declared in the calling program. It is not altered on exit.

N integer variable, which on entry, the user must set to the actual size of matrix A which must be less than or equal to the declared dimensions of A in the calling program. It is not altered on exit.

A real two-dimensional array of size NxM ($M \geq N$) an entry, the array A specifies the matrix to be decomposed; on exit A is overwritten with its decomposition.

SC real one-dimensional array of size N. On exit, it contains the scaling factors of matrix A.

L integer one-dimensional array of size N. On exit, array L contains the record of row interchanges during decomposition. L(1) = 2 means that row 2 has been moved to row 1.

IFAIL integer variable, it is set to zero on succesful exit from RF1LE, otherwise it is set to a non-zero value.

Note: All real variables must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics.

A successful return from RF1LE is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	SYSTEM IS SINGULAR	One of the rows or columns is zero or two rows (columns) are linearly dependent-change matrix.

2.	SYSTEM IS NEARLY SINGULAR	Exit from the subroutine is not forced by this error criteria. Computation is continued to the end. It is possible to use the decomposition but results may not be correct.

f) Other routines used

Function IFlMQ is referenced.

g) Timing

Depends on the size of array A. (It is proportional to $N^3/_3$).

h) Storage

There are no internally declared arrays and the compiled output of RFlLE using Tx 990/4 microcomputer is 1,362 bytes.

i) Accuracy

For well-conditioned systems of moderate size (below 20 x 20), the results produced are accurate to near machine precision.

j) Example

The usage of RFlLE is best explained when it is used with other subroutines.

2. SUBROUTINE RF2LE

a) Purpose

The subroutine RF2LE solves a system of simultaneous linear equations after matrix A has been decomposed.

b) Specification

The subroutine is written in FORTRAN and it is of the form.

SUBROUTINE RF2LE (ID, N, A, B, X, SC, L)

c) Description

This subroutine solves the system

$$LUx = b$$

where LU is the decomposition of A by subroutine RF1LE. RF2LE can only be used after A has been replaced by its decomposition. RF2LE obtains x by solving:

$$Lc = b$$

and

$$Ux = c$$

d) Parameters

ID — integer variable, on entry, the user must set it to the row size of matrix A declared in the calling program. It is not altered on exit.

N — integer variable which on entry, the user must set to the actual size of matrix A which should not be greater than the declared dimensions of A in the calling program. It is not altered on exit.

A real two-dimensional array of size N x M ($M \geq N$).
On entry, array A Specifies the matrix overwritten by its decomposition. It is unaltered on exit.

B real one-dimensional array of size N.
On entry B specifies the right hand side of the equation Ax = b. It is not altered on exit.

X real one-dimensional array of size N. On exit X contains the solution of the system.

SC real one-dimensional array of size N.
On entry SC contains the scaling factor of the original matrix A obtained during decomposition. It is not altered on exit.

L integer one-dimensional array of size N.
On entry L contains the record of row interchanges obtained during decomposition. It is unaltered on exit.

NOTE: All real variables must be declared as DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

None.

f) Other routines used.

None.

g) Timing

Depends on the number of equations to be solved and it

is approximately proportional to n(n+1). For the example given, RF2LE took .056 sec and decomposition (RF1LE) took 0.16 sec., using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF2LE using TX990/4 microcomputer is 690 bytes.

i) Accuracy

For well-conditioned systems of moderate size (below 20) the results produced are accurate to near machine precision.

j) Example

RF2LE was used to solve the system of linear equations:

$$\begin{bmatrix} 0.96 & -.08 & -.12 & -.16 & 0.04 \\ -.08 & 0.84 & -.24 & -.32 & 0.08 \\ -.12 & -.24 & 0.64 & -.48 & 0.12 \\ -.16 & -.32 & -.48 & 0.36 & 0.16 \\ 0.04 & 0.08 & 0.12 & 0.16 & -.04 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0.64 \\ 0.28 \\ -.08 \\ -.44 \\ 0.36 \end{bmatrix}$$

after RF1LE has been used to decompose the system. Example B1 shows how RF1LE and RF2LE can be used together to solve a system of linear equations simultaneously. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

EXAMPLE B1

```
      REAL A(10,10),B(10),X(10),SC(10)
      INTEGER L(10)
      DATA IN,IOUT/20,21/
C  INPUT NUMBER OF EQUATIONS
      READ(IN,100) N
C  INPUT MATRIX A
      READ(IN,200)((A(I,J),J=1,N),I=1,N)
C  INPUT MATRIX B
      READ(IN,200)(B(I),I=1,N)
C  DECOMPOSE A FIRST
      CALL RF1LE(10,N,A,SC,L,IFAIL)
      IF(IFAIL.NE.0) STOP
C  SOLVE AX=B
      CALL RF2LE(10,N,A,B,X,SC,L)
      WRITE(IOUT,200)(X(I),I=1,N)
100   FORMAT(I4)
200   FORMAT(5F5.2)
      STOP
      END


          DATA


   5
 0.96 -.08 -.12 -.16 0.04
 -.08 0.84 -.24 -.32 0.08
 -.12 -.24 0.64 -.48 0.12
 -.16 -.32 -.48 0.36 0.16
 0.04 0.08 0.12 0.16 -.04
 0.64 0.28 -.08 -.44 0.36


          RESULTS

 1.00 1.00 1.00 1.00 1.00
```

3. SUBROUTINE RF3LE

a) Purpose

Subroutine RF3LE is used to refine the solution obtained by RF1LE and RF2LE.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

```
SUBROUTINE RF3LE (ID,N,AB,A,B,X,SC,RES,WS2,L,IT)
```

c) Description

RF3LE improves the solution computed by RF1LE and RF2LE. By substituting a given solution, using partial double precision arithmetic, residual r given by

$$r = b - Ax$$

is computed. The equation

$$Ae = r$$

is solved and the original solution x is updated by x + e. Before RF3LE is used, the matrix A must have been decomposed and approximate solution obtained.

d) Parameters

ID — integer variable, which on entry, must be set to the row size matrix A declared in the calling program. It is not altered on exit

N — integer variable which on entry the user must set to the actual size of matrix A. N should not be greater than the declared dimensions of A in the calling program. It is not altered on exit.

AB — real two-dimensional array of size NXM ($M \geq N$) which on entry contains the undecomposed matrix

A. In the calling program ID should be the same for AB and A. On exit AB is not altered.

A — real two-dimensional array of size NXM ($M \geq N$) which on entry specifies the matrix overwritten by its decomposition. A is not altered on exit.

B — real one-dimensional array of size N. On entry B specifies the right hand side of the equation $Ax = b$. It is not altered on exit.

X — real one-dimensional array of size N. On entry X contains an approximate solution of $Ax = b$. On exit, X contains an improved solution of the system $Ax=b$.

SC — real one-dimensional array of size N. On entry SC contains the scaling factors of the original matrix A obtained during the decomposition of A. It is not altered on exit.

L — integer one-dimensional array of size N. On entry L contains the record of row interchanges obtained during decomposition. It is unaltered on exit.

RES — real one-dimensional array of size N, used as workspace.

WS2 — real one-dimensional array of size N, used as workspace.

IT — integer variable which on entry the user must set to the number of refinements required. The value 2 is suggested.

e) Error diagnostics

None

f) Other routines used

None

g) Timing

Depends on the size of AB. Much time is taken up by the computation of the residual.

h) Storage

No internally declared arrays and the compiled output of RF3LE using Tx990/4 microcomputer is 1,242.

i) Accuracy

RF3LE is designed to increase the accuracy of a given solution.

j) Example

None

k) Further comments

This subroutine is written in partial double precision. It can therefore only be implemented if double precision arithmetic is available. It is only useful if the single precision version of the library is implemented.

4. SUBROUTINE RF4LE

a) Purpose

The subroutines RF4LE solves tridiagonal linear systems.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

SUBROUTINE RF4LE (A,B,C,D,N,IFAIL)

c) Description.

The subroutine uses Gaussian elimination without pivoting and it is a FORTRAN version of Osterby (125) Algol 60

procedure. However in RF4LE a test for singularity is carried out.

d) <u>Parameters</u>

A real one-dimensional array of size N. On entry, it forms the lower diagonal of the tridiagonal matrix, and its values are A(I), I=2,.....,N. It is not altered on exit.

B real one-dimensional array of size N. On entry, it forms the diagonal of the tridiagonal matrix

C real one-dimensional array of size N. On entry, it forms the upper diagonal of the tridiagonal matrix and its values are C(I), I=2, ...,N. It is not altered on exit.

D real one-dimensional array of size N. On entry, it is the right hand side of the tridiagonal system:

$$\begin{bmatrix} b_1 & c_2 & & & \\ a_2 & b_2 & c_3 & & \\ & & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & a_n & b_n \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_n \end{bmatrix}$$

On exit it contains the solution of the system.

N integer variable. On entry, the user must set it to the size of the system. It is not altered on exit.

IFAIL integer variable and it is set to zero on successful exit from RF4LE, otherwise it is set to a non zero value.

NOTE: All real variables must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

A successful return from RF4LE is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	SYSTEM IS SINGULAR	Change system.

f) Other routines used

Function IF1MQ is referenced.

g) Timing

Depends on the size of the system and it is approximately proportional to 5n-4. For the example given, RF4LE took .042 sec using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF4LE using Tx990/4 microcomputer is 648 bytes.

i) Accuracy

If the coefficient matrix is diagonally dominant ie $|b_i| \geq |a_i| + |c_{i+1}|$, then the result produced is very likely to be of high accuracy (reliable).

j) Example

RF4LE was used to solve the tridiagonal linear system:

$$\begin{bmatrix} 2.1 & 1.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 3.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.8 & 4.0 & 2.0 & 0.0 \\ 0.0 & 0.0 & 2.0 & 5.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 1.8 & 2.2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3.1 \\ 4.5 \\ 6.8 \\ 8.0 \\ 4.0 \end{bmatrix}$$

Example B2 shows how RF4LE can be used to solve a tridiagonal linear system. Note how the arrays A, B, C, D are assigned values in the example. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

EXAMPLE 32

```
      REAL A(10),B(10),C(10),D(10)
      DATA IN,IOUT/20,21/
C  INPUT NUMBER OF EQUATIONS
      READ(IN,100) N
C  INPUT LOWER DIAGONAL
      READ(IN,300)(A(I),I=2,N)
C  INPUT DIAGONAL
      READ(IN,200)(B(I),I=1,N)
C  INPUT UPPER DIAGONAL
      READ(IN,300)(C(I),I=2,N)
C  INPUT COLUMN MATRIX D
      READ(IN,200)(D(I),I=1,N)
C  CALL SUBROUTINE
      CALL RF4LE(A,B,C,D,N,IFAIL)
      IF(IFAIL.NE.0) STOP
C  SOLUTION IS IN D
      WRITE(IOUT,200)(D(I),I=1,N)
100   FORMAT(I4)
200   FORMAT(5F5.2)
300   FORMAT(4F5.2)
      STOP
      END
```

DATA

```
  5
 1.00 0.80 2.00 1.80
 2.10 3.00 4.00 5.00 2.20
 1.00 0.50 2.00 1.00
 3.10 4.50 6.80 8.00 4.00
```

RESULT

```
 1.00 1.00 1.00 1.00 1.00
```

5. SUBROUTINE RF1IN

a) Purpose

The subroutine RF1IN computes the inverse of a nonsingular n x n matrix after it has been decomposed using RF1LE.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

SUBROUTINE RF1IN (ID, N, A, AINV, SC, LP)

c) Description

This subroutine computes the inverse of an nxn nonsingular matrix by solving the systems of linear equations

$$Ax = e_j \quad j = 1, \ldots\ldots, n$$

where A is the matrix whose inverse is required and each ej is a unit vector. Each x is a column vector of the inverse of A. (see Johnson et al (93) and Forsythe et al (59)).

d) Parameters

ID — integer variable, which on entry, must be set to the row size of matrix A declared in the calling program. It is not altered on exit.

N — integer variable which on entry the user must set to the actual size of matrix A. N must not be greater than the declared dimensions of A in the calling program. It is not altered on exit.

A — real two-dimensional array of size NxM ($M \geq N$), which on entry specifies the already decomposed form of the matrix whose inverse is required. It is unaltered on exit.

AINV — real two-dimensional array of size NxM ($M \geq N$), which on exit contains the inverse.

SC real one-dimensional array of size N. On entry SC specifies the scaling factors of the original matrix A obtained during the decomposition A. It is not altered on exit.

LF One-dimensional integer array of size N. On entry LF specifies the record of row interchanges obtained during decomposition. It is unaltered on exit.

NOTE: All real variables must be declared as DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

None

f) Other routines used

None

g) Timing

Depends on the size of array A. For the example given, RF1IN took 0.24 sec using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF1IN using Tx990/4 microcomputer is 648 bytes.

i) Accuracy

For well conditioned matrices of moderate sizes (below 20) the results produced are accurate to near machines precision.

j) Example

RF1IN was used to compute the inverse of the matrix given in the DATA section of the example. Note that the matrix must

EXAMPLE B3

```
      REAL A(10,10),AINV(10,10),X(10),SC(10)
      INTEGER L(10)
      DATA IN,IOUT/20,21/
C  INPUT SIZE OF MATRIX
      READ(IN,100) N
C  INPUT MATRIX A
      READ(IN,200)((A(I,J),J=1,N),I=1,N)
C  DECOMPOSE A FIRST
      CALL RF1LE(10,N,A,SC,L,IFAIL)
      IF(IFAIL.NE.0) STOP
C  OBTAIN INVERSE
      CALL RF1IN(10,N,A,AINV,SC,L)
      WRITE(IOUT,200)((AINV(I,J),J=1,N),I=1,N)
100   FORMAT(I4)
200   FORMAT(5F5.2)
      STOP
      END
```

DATA

```
  5
0.96 -.08 -.12 -.16 0.04
-.08 0.84 -.24 -.32 0.08
-.12 -.24 0.64 -.48 0.12
-.16 -.32 -.48 0.36 0.16
0.04 0.08 0.12 0.16 -.04
0.64 0.28 -.08 -.44 0.36
```

RESULTS

```
1.00 0.00 0.00 0.00 1.00
0.00 1.00 0.00 0.00 2.00
0.00 0.00 1.00 0.00 3.00
0.00 0.00 0.00 1.00 4.00
1.00 2.00 3.00 4.00 5.00
```

be decomposed first before RF1IN is applied. This example (B3) which is in single precision, requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

6. SUBROUTINE RF1DE

a) Purpose

The subroutine RF1DE calculates the determinant of a given nxn matrix after it has been decomposed.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

SUBROUTINE RF1DE (ID, N, A, SC, DET, L, IFAIL)

c) Description

After a matrix has been decomposed by the use of subroutine RF1LE, subroutine RF1DE computes the determinant by summing up the logarithms of the diagonal elements of the upper triangle of the decomposition and obtaining the antilogarithm of this sum. In a situation where the solution is out of machine range, the solution is left in logarithmic form (see Forsythe et al (59))

d) Parameters

ID — integer variable, which on entry, the user must set to the row size of matrix A declared in the calling program. It is not altered on exit.

N — integer variable which on entry the user must set to the actual size of matrix A and it must be less than or equal to the declared dimensions

of A in the calling program. It is not altered on exit.

A — real two-dimensional array of size NXM ($M \geq N$) On entry, array A specifies the matrix overwritten with its decomposition. It is not altered on exit.

SC — real one-dimensional array of size N. On entry SC contains the scaling factors of the original matrix A, obtained during decomposition. It is not altered on exit.

DET — real variable which on exit contains the determinant of the matrix

L — integer one-dimensional matrix of size N which on entry to the subroutine, contains the record of row interchanges obtained during decomposition. It is unaltered on exit.

IFAIL — integer variable, it is set to zero on successful exit from RF1DE, otherwise it is set to a non-zero value. In such a situation IFAIL contains the sign of the determinant.

<u>NOTE</u>: All real variables must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

IFAIL	ERROR MESSAGES	RECOVERY
±1	LOG10 OF DET PROVIDED ON EXIT.	This means that it has been determined that the determinant will be out of range and as a result on exit, the $\log_{10}$ of the determinant is provided. IFAIL contains the sign of the determinant.

f) Other routines used

The function IF1MQ is referenced.

g) Timing

Depends on the size of matrix A and for the example given RF1DE took 0.048 using Tx990/4 microcomputer excluding time for decomposing the matrix.

h) Storage

There are no internally declared arrays and the compiled output of RF1DE using Tx990/4 microcomputer is 690 bytes.

i) Accuracy

For well conditioned matrices of moderate sizes (below 20) the results produced are accurate to near machine precision.

j) Example

RF1DE was used to compute the determinant of a matrix given in the DATA section of example B4. Note that the matrix must be decomposed first before RF1DE is applied. Example B4 which is in single precision, requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

EXAMPLE B4

```
      REAL A(10,10),SC(10)
      INTEGER L(10)
      DATA IN,IOUT/20,21/
C  INPUT SIZE OF MATRIX
      READ(IN,100) N
C  INPUT MATRIX A
      READ(IN,200)((A(I,J),J=1,N),I=1,N)
C  DECOMPOSE A FIRST
      CALL RF1LE(10,N,A,SC,L,IFAIL)
      IF(IFAIL.NE.0) STOP
C  OBTAIN DETERMINANT
      CALL RF1DE(10,N,A,SC,DET,L,IFAIL)
      WRITE(IOUT,300) DET
300   FORMAT(6X,E14.7)
100   FORMAT(I4)
200   FORMAT(5F5.2)
      STOP
      END


        DATA

   5
 0.96 -.08 -.12 -.16 0.04
 -.08 0.84 -.24 -.32 0.08
 -.12 -.24 0.64 -.48 0.12
 -.16 -.32 -.48 0.36 0.16
 0.04 0.08 0.12 0.16 -.04
 0.64 0.28 -.08 -.44 0.36


        RESULT


     -0.3999994E-01
```

SUBROUTINE RF1LE


```
      SUBROUTINE RF1LE(ID,N,A,SC,L,IFAIL)
C  THIS ROUTINE DECOMPOSES MATIX A(ID,N) IN PLACE USING
C  CROUT METHOD.
C.    DOUBLE PRECISION A,SC,Q,TL,S,DF2MQ
C-    DOUBLE PRECISION P
      DIMENSION A(ID,N),SC(N),L(N)
C   L(1)=2 MEANS THAT ROW 2 HAS BEEN MOVED TO ROW 1
      IOUT=IF1MQ(1)
      TL=RF2MQ(0.1)
C*    TL=DF2MQ(0.0)
      IFAIL=0
      NM1=N-1
C
C   INTIALIZE POINTERS
C
      DO 10 I=1,N
10    L(I)=I
C
C   SCALE THE ELEMENTS OF ARRAY A AND PRESERVE THE SCALING FACTORS
C
      DO 15 I=1,N
      Q=0.0E 00
      DO 25 J=1,N
      S=ABS(A(I,J))
C*    S=DABS(A(I,J))
      IF(Q.GE.S) GO TO 25
      Q=S
25    CONTINUE
      IF(Q .EQ. 0.0E 00) GO TO 85
      Q=0.10E 01/Q
      DO 35 K=1,N
35    A(I,K)=Q*A(I,K)
15    SC(I)=Q
C
C   SEARCH FOR HIGHEST PIVOT ELEMENT
C
      DO 20 K=1,NM1
      Q=0.0E 00
      DO 30 I=K,N
      S=ABS(A(I,K))
C*    S=DABS(A(I,K))
      IF(Q.GE.S) GO TO 30
      Q=S
      J=I
30    CONTINUE
C
C  TEST FOR SINGULARITY AND NEAR SINGULARITY
C
      IF(Q .EQ. 0.0E 00) GO TO 85
      IF(Q .GT. TL) GO TO 45
```

RF1LE Cont.


```
      IFAIL=2
45    IF(J .EQ. K) GO TO 40
C
C  INTERCHANGE ROWS
C
      I=L(K)
      L(K)=L(J)
      L(J)=I
      DO 50 I=1,N
      Q=A(K,I)
      A(K,I)=A(J,I)
50    A(J,I)=Q
40    Q=0.10E 01/A(K,K)
      KP1=K+1
C
C  PERFORM DECOMPOSITION
C
      DO 60 I=KP1,N
60    A(I,K)=Q*A(I,K)
      DO 70 I=2,N
      M1=I-1
      P=0.00E 00
      IF(M1.GT.K) M1=K
      DO 80 J=1,M1
      P=P+A(I,J)*A(J,KP1)
C+    P=P+DBLE(A(I,J))*A(J,KP1)
80    CONTINUE
70    A(I,KP1)=A(I,KP1)-P
C+ 70 A(I,KP1)=SNGL(A(I,KP1)-P)
20    CONTINUE
C
C  TEST FOR SINGULARITY AND NEAR SINGULARITY
C
      IF(A(N,N).NE.0.0) GO TO 90
85    IFAIL=1
      WRITE(IOUT,100) IFAIL
100   FORMAT(4X,I4,4X,18HSYSTEM IS SINGULAR/)
      RETURN
90    IF((IFAIL.EQ.0).AND.(ABS(A(N,N)).GT.TL)) RETURN
C* 90 IF((IFAIL.EQ.0).AND.(DABS(A(N,N)).GT.TL)) RETURN
      IFAIL=2
      WRITE(IOUT,200) IFAIL
200   FORMAT(4X,I4,4X,25HSYSTEM IS NEARLY SINGULAR/)
      RETURN
      END
```

SUBROUTINE RF2LE


```
      SUBROUTINE RF2LE(ID,N,A,B,X,SC,L)
C     DOUBLE PRESISION A,B,X,SC
C-    DOUBLE PRECISION Q
      DIMENSION A(ID,N),B(N),X(N),SC(N),L(N)
      NP1=N+1
      DO 10 I=1,N
      J=L(I)
10    X(I)=SC(J)*B(J)
C
C  FORWARD SUBSTITUTION
C
      NM1=N-1
      DO 20 I=2,N
      IM1=I-1
      Q=0.0E 00
      DO 30 J=1,IM1
30    Q=Q+A(I,J)*X(J)
C+ 30 Q=Q+DBLE(A(I,J))*X(J)
20    X(I)=X(I)-Q
C+ 20 X(I)=SNGL(X(I)-Q)
C
C  BACKWARD SUBSTITUTION
C
      X(N)=X(N)/A(N,N)
      DO 40 K=1,NM1
      Q=0.0E 00
      NMK=N-K
      DO 50 J=1,K
      I=NP1-J
50    Q=Q+A(NMK,I)*X(I)
C+ 50 Q=Q+DBLE(X(I))*A(NMK,I)
40    X(NMK)=(X(NMK)-Q)/A(NMK,NMK)
C+ 40 X(NMK)=SNGL((X(NMK)-Q)/A(NMK,NMK))
      RETURN
      END
```

SUBROUTINE RF3LE


```
      SUBROUTINE RF3LE(ID,N,AB,A,B,X,SC,RES,WS2,L,IT)
C  THIS ROUTINE REFINES THE SOLUTION X WHERE AB AND A ARE THE ORIGINAL
C  AND DECOMPOSED MATRICES OF SIZE NxN RESPECTIVELY
      DOUBLE PRECISION Q
      DIMENSION AB(ID,N),A(ID,N),B(N),X(N),RES(N),WS2(N),SC(N),L(N)
      NP1=N+1
      NM1=N-1
      DO 110 M=1,IT
C
C  CALCULATE THE RESIDUALS RES
C
      DO 20 I=1,N
      Q=0.00E 00
      DO 10 J=1,N
10    Q=Q+DBLE(X(J))*AB(I,J)
20    RES(I)=SNGL(B(I)-Q)
C
C  SCALE THE RESIDUALS RES
C
      DO 30 I=1,N
30    RES(I)=SC(I)*RES(I)
C
C  SOLVE FOR THE ERROR IN X
C
40    DO 50 I=1,N
      J=L(I)
50    WS2(I)=RES(J)
      DO 70 I=2,N
      IM1=I-1
      Q=0.00E 00
      DO 60 J=1,IM1
60    Q=Q+A(I,J)*DBLE(WS2(J))
70    WS2(I)=SNGL(WS2(I)-Q)
      WS2(N)=WS2(N)/A(N,N)
      DO 90 K=1,NM1
      Q=0.00E 00
      NMK=N-K
      DO 80 J=1,K
      I=NP1-J
80    Q=Q+A(NMK,I)*DBLE(WS2(I))
90    WS2(NMK)=SNGL((WS2(NMK)-Q)/A(NMK,NMK))
C
C  IMPROVE X BY ADDING ERROR WS2
C
      DO 100 I=1,N
100   X(I)=X(I)+WS2(I)
110   CONTINUE
      RETURN
      END
```

SUBROUTINE RF4LE

PAGE 1

```
      SUBROUTINE RF4LE(A,B,C,D,N,IFAIL)
C  SOLUTION OF TRIDIAGONAL SYSTEM USING OSTERBY VERSION OF
C  GAUSSIAN ELIMINATION
C.    DOUBLE PRECISION A,B,C,D,BI,DI,Z
      DIMENSION A(N),B(N),C(N),D(N)
      IOUT=IF1MQ(1)
      BI=B(1)
      DI=D(1)
C  DECOMPOSE
      DO 10 I=2,N
      IF(BI.EQ.0.00E 00) GO TO 30
      Z=A(I)/BI
      BI=B(I)-C(I)*Z
      B(I)=BI
      DI=D(I)-DI*Z
10    D(I)=DI
C BACK SUBSTITUTION
      IF(B(N).EQ.0.0E 00) GO TO 30
      DI=D(N)/B(N)
      D(N)=DI
      NM1=N-1
      DO 20 J=1,NM1
      I=N-J
      IF(B(I).EQ.0.00E 00) GO TO 30
      DI=(D(I)-C(I+1)*DI)/B(I)
20    D(I)=DI
      IFAIL=0
      RETURN
30    IFAIL=1
      WRITE(IOUT,100)IFAIL
100   FORMAT(4X,I4,4X,18HSYSTEM IS SINGULAR/)
      RETURN
      END
```

SUBROUTINE RF1IN


```
      SUBROUTINE RF1IN(ID,N,A,AINV,SC,LP)
C
C  THIS ROUTINE COMPUTE THE INVERSE OF A AND STORE IT IN AINV
C  A MUST BE DECOMPOSED INITIALLY.
C
C-    DOUBLE PRECISION Q
C.    DOUBLE PRECISION A,AINV,SC
      DIMENSION A(ID,N),AINV(ID,N),SC(N),LP(N)
      NP1=N+1
      NM1=N-1
C
C  THE COLUMNS OF AINV ARE NOW BEING COMPUTED
C
      DO 70 L=1,N
      DO 80 J=1,N
80    AINV(J,L)=0.0
      DO 10 I=1,N
      J=LP(I)
      IF(J .NE. L) GO TO 10
      AINV(I,L)=SC(J)
      GO TO 90
10    CONTINUE
90    DO 20 I=2,N
      IM1=I-1
      Q=0.0E 00
      DO 30 J=1,IM1
30    Q=Q+A(I,J)*AINV(J,L)
C+ 30 Q=Q+DBLE(A(I,J))*AINV(J,L)
20    AINV(I,L)=AINV(I,L)-Q
C+ 20 AINV(I,L)=SNGL(AINV(I,L)-Q)
      AINV(N,L)=AINV(N,L)/A(N,N)
      DO 40 K=1,NM1
      Q=0.0E 00
      NMK=N-K
      DO 50 J=1,K
      I=NP1-J
50    Q=Q+A(NMK,I)*AINV(I,L)
C+ 50 Q=Q+DBLE(A(NMK,I))*AINV(I,L)
40    AINV(NMK,L)=(AINV(NMK,L)-Q)/A(NMK,NMK)
C+ 40 AINV(NMK,L)=SNGL((AINV(NMK,L)-Q)/A(NMK,NMK))
70    CONTINUE
      RETURN
      END
```

SUBROUTINE RF1DE


```
      SUBROUTINE RF1DE(ID,N,A,SC,DET,L,IFAIL)
C
C  THIS ROUTINE COMPUTES THE DETERMINANT OF A AND STORES IT
C  IN DET. A IS NOT DECOMPOSED INITIALLY.
C
C-    DOUBLE PRECISION Q
C.    DOUBLE PRECISION A,SC,P,TEN,DET
      DIMENSION A(ID,N),SC(N),L(N)
      IFAIL=0
      IOUT=IF1MQ(1)
      DET=0.0E 00
      TEN=0.10000E 02
      K=1
      Q=0.0E 00
C
C  FIND DETERMINANT DET
C
      DO 10 I=1,N
      P=A(I,I)
      IF(P .EQ. 0.0E 00) RETURN
      P=P/SC(I)
      IF(P .GT. 0.0E 00) GO TO 10
      K=-K
      P=-P
10    Q=Q+ALOG10(P)
C* 10 Q=Q+DLOG10(P)
      IU=-1
      DO 20 I=1,N
      IP=L(I)
      IF(IP.EQ.I) GO TO 20
      IU=IU+1
20    CONTINUE
      IF(IU.EQ.-1) GO TO 30
      IU=IU-2*(IU/2)
      IF(IU .NE. 0) K=-K
30    ALOGD=Q
C+30  ALOGD=SNGL(Q)
      I=IF1MQ(4)
C*    P=DBLE(FLOAT(I))
      IP=IF1MQ(5)
      P=FLOAT(I)
      S=FLOAT(IP)
C*    S=DBLE(FLOAT(IP))
      IF((S.LT.ALOGD).AND.(ALOGD.LT.P)) GO TO 40
      IFAIL=K
      DET=ALOGD
      WRITE(IOUT,100) IFAIL
100   FORMAT(4X,I4,4X,29HLOG10 OF DET PROVIDED ON EXIT/)
      RETURN
40    DET=TEN**ALOGD
```

RF1DE Cont.

PAGE 2

```
51      IF(K .LT. 0) DET=-DET
52      RETURN
53      END
```

APPENDIX C:

ROOTS OF NON-LINEAR FUNCTIONS

The subroutines documented in this appendix are used to determine the roots of non-linear single variable functions. If the function is a polynomial then all the roots are obtained by a single call to a subroutine, otherwise only one root is obtained. Table C summarizes the actions of the subroutines.

SUBROUTINE NAME	PURPOSE
RF1RF	This subroutine searches for an interval containing a root if this is not given by the user. It then applies Brent's method once an interval containing a root is known.
RF1ZP	This subroutine determines all the roots of a real polynomial (using Grant's and Hitchin's algorithm)

TABLE C: Summary of subroutines for finding roots.

1. SUBROUTINE RF1RF

a) Purpose

RF1RE determines a root of a non linear, single variable function in a given interval. It can also search for an interval containing a root.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

```
SUBROUTINE RF1RF (DA,DB,ROOT,F,IT,IFAIL)
```

c) Description

Given an interval (a,b), a test for a root in (a,b) is carried out by checking whether f(a) and f(b) have different signs, (where f is the function whose root is to be determined). If there is no root in (a,b), a search algorithm is used to determine such an interval. Brent's algorithm which combines, the bracking property of bisection method with the superlinear convergence property of linear interpolation is then applied. (see Brent (14) for details)

d) Parameters

DA real variable which on entry, the user must set to the left endpoint of the initial interval. It is unaltered on exit.

DB real variable which on entry, the user must set to the right endpoint of the initial interval. It is unaltered on exit.

F real function whose root is to be determined and it is supplied by the user. It's specification is:

```
REAL FUNCTION  F(X)
REAL X
```

F must be declared as External in the (sub) program from which it is called.

IT integer variable which on entry, the user must

set to the maximum number of function evaluations necessary to locate an interval containing a root. If a user is sure that the supplied interval (DA, DB) contains a root (ie F(DA) x F(DB) < 0), then IT can be given any value on entry. IT is not altered on exit.

IFAIL integer variable, which on successful exit, is set to zero, otherwise it is set to a non-zero value.

NOTE: All real variables and functions must be specified as DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

A successful return from RF1RF is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	AN INTERVAL CONTAINING A ROOT NOT LOCATED	Use another interval or increase the number of function evaluations allowed.

f) Other routines used

Function IF1MQ is referenced.

g) Timing

Depends on the function whose root is required.

h) Storage

There are no internally declared arrays and the compiled

output of RF1RF using Tx990/4 microcomputer occupies 1,742 bytes of storage.

i) Accuracy

The result produced is accurate to 2 x machine relative error.

j) Example

Example C1 shows how RF1RF can be used to obtain a root of a function in a given interval. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

2. SUBROUTINE RF1ZP

a) Purpose

Subroutine RF1ZP computes the roots of a real polynomial to machine precision.

b) Specification

RF1ZP is of the form:

SUBROUTINE RF1ZP (A, REZ, AIMZ, B, C, NP1, IFAIL)

c) Description

Let $f(z) = a_o z^n + a_1 z^{n-1} + \ldots\ldots + a_n$ 2.1

be the polynomial whose roots are required. This method obtains a root of f(z) by minimising

$$\phi(Z) = R^2(x,y) + J^2(x,y) \qquad 2.2$$

where $f(z) = f(x+iy) = R(x,y) + iJ(x,y)$ and R,J, are real functions with $i = \sqrt{-1}$

The root obtained is factored out of f(z) using composite deflation. The process is continued until n = 2 or 1.

The remaining roots are then obtained by solving a quadratic

EXAMPLE C1

```
0001        REAL ROOT,A,B
0002        EXTERNAL F
0003        DATA INPUT,IOUT/1,2/
0004 C  DEFINE AN INTERVAL [A,B] WHICH MAY  CONTAIN A ROOT
0005        A=2.0
0006        B=3.0
0007 C  SPECIFY THE NUMBER OF FUNCTION EVALUATIONS NECESSARY TO
0008 C  LOCATE AN INTERVAL CONTAINING A ROOT (1000 IS ENOUGH)
0009        IT=1000
0010        CALL RF1RF(A,B,ROOT,F,IT,IFAIL)
0011        IF(IFAIL.NE.0) STOP
0012        WRITE(IOUT,100) ROOT
0013 100    FORMAT(8X,5HROOT=,E14.7/)
0014        STOP
0015        END
0016
0017
0018        REAL FUNCTION F(X)
0019        F=X**3.0-2.0*X-5.0
0020        RETURN
0021        END
0022
0023
0024
0025            RESULT
0026
0027
0028          ROOT= 0.2094551E 01
```

equation or solving ax + b = 0. (see Grant et al (77)).

d) Parameters

A	real one-dimensional array of size NP1 which on entry contains the coefficients of the polynomial whose roots are to be determined. A(1) holds the Coefficient of the highest power of X. (A(1) * X * * N)
REZ	real one-dimensional array of size NP1. On exit REZ(1), ..., REZ(NP1-1) contain the real parts of the roots of the Polynomial.
AIMZ	real one-dimensional array of size NP1. On exit AIMZ(1), ..., AIMZ (NP1-1) hold the corresponding imaginary parts of the roots of the polynomial.
B	real one-dimensional array of size NP1 used as workspace.
C	real one-dimensional array of size NP1 used as workspace.
NP1	integer variable which on entry the user must set to the number of coefficients of the polynomial. (degree of polynomial + 1)
IFAIL	integer variable which is set to zero on successful exit, otherwise it is set to a non-zero value.
Note:	All real variables and functions must be specified as DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is unaltered by the subroutine can be replaced with a constant.

e) Error diagnostics:

A successful return from RF1ZP is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	FIRST COEFF IS 0.0 OR NUM OF COEFF LT 2	This means that A(1) = 0.0 or NP1< 2. Check these variables and make appropriate change.
2	CANNOT ESCAPE FROM SADDLE POINT	No recovery, but this situation will hardly arise even for $Z^n + 1 = 0$ since the given prefixed values usually result in obtaining a root which is a saddle point. NP1 contains the number of computed roots before saddle point was detected and REZ(K), AIMZ(K), K = 1 ...., NP1 contain the determined roots.

f) Other routines used

Subroutines RF3ZP, RF5ZP, RF6ZP and RF7ZP are used only by RF1ZP. Infact they are part of RF1ZP and the user is not aware of their usage. These subroutines are not available to the user. Function IF1MQ, is also referenced by subroutine RF1ZP.

g) Timing:

Depends on the degree of the polynomial, and the roots of the polynomial. For the example given, it took Tx990/4 0.9 sec to compute the six roots.

h) Storage:

There are no internally declared arrays but there is LABEL COMMON storage. The compiled output of RF1ZP together with RF3ZP, RF5ZP, RF6ZP and RF7ZP using Tx990/4 microcomputer, occupies 5,872 bytes of storage. In the library this algorithm requires the maximum amount of storage.

i) Accuracy

Results produced are accurate to machine precision except for multiple roots and other ill-conditioned polynomials.

j) Example

Example C2 shows how RF1ZP can be used to obtain all the roots of polynomial. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

EXAMPLE C2

```
      REAL A,B,C,REZ,AIMZ
      DIMENSION A(20),B(20),C(20),REZ(20),AIMZ(20)
      DATA IN,IOUT/20,21/
C
C  INPUT DEGREE OF POLYNOMIAL
C
      READ(IN,100) N
      NP1=N+1
C
C  INPUT THE COEFFICIENTS OF THE POLYNOMIAL WITH
C  A(1) CORRESPONDING TO THE COEFFIENT OF X**N
C
      READ(IN,200)(A(K),K=1,NP1)
C  CALL SUBROUTINE RP1ZP
      CALL RP1ZP(A,REZ,AIMZ,B,C,NP1,IFAIL)
C  ARRAY REZ AND AIMZ CONTAIN THE REAL AND IMAGINARY PARTS
C  OF THE ROOTS RESPECTIVELY
C  ARRAYS B AND C ARE USED AS WORKSPACE
C
      IF(IFAIL.NE.0) STOP
C  PRINT OUT RESULT
      WRITE(IOUT,300)(REZ(K),AIMZ(K),K=1,N)
100   FORMAT(I4)
200   FORMAT(5E14.7)
300   FORMAT(2E14.7/)
      STOP
      END
```

DATA

```
   6
 0.1000000E 01 -.2000000E 01 0.5000000E 01 -.6000000E 01 0.2000000E 01
 0.3000000E 01 -.4000000E 01
```

RESULTS

```
 0.1192093E-05 0.1999999E 01
 0.1192093E-05 -.1999999E 01
 -.1000001E 01 0.0000000E 00
 0.9999994E 00 0.9999995E 00
 0.9999994E 00 -.9999995E 00
 0.1000000E 01 0.0000000E 00
```

SUBROUTINE RF1ZP


```
      SUBROUTINE RF1RF(DA,DB,ROOT,F,IT,IFAIL)
C  ROUTINE RF1RF RETURNS A ROOT OF THE FUNCTION F WHICH MAY
C  NOT BE IN THE GIVEN INTERVAL [DA,DB] TO WITHIN DOUBLE
C  MACHINE PRECISION,USING SEARCH AND BRENT'S METHOD.
C.    DOUBLE PRECISION A,B,C,FA,FC,FB,Q,P,Q2,Q3,EPS,E,RM,F,D,S,
C.   *                 R,TL,TL2,DF2MQ,ROOT,H,DA,DB
      IFAIL=1
      A=DA
      B=DB
      IC=0
      FA=F(A)
      FB=F(B)
      H=0.10E-03
      IF((FA.GT.0.0E 00).AND.(FB.LT.0.0E 00)) GO TO 9
      IF((FA.LT.0.0E 00).AND.(FB.GT.0.0E 00)) GO TO 9
      B=A
C  SEARCH FOR AN INTERVAL CONTAINING A ROOT
      IOUT=IF1MQ(1)
5     IC=IC+1
      IF(IC.LE.IT) GO TO 6
      WRITE(IOUT,600) IFAIL
      RETURN
600   FORMAT(I4,2X,41HAN INTERVAL CONTAINING A ROOT NOT LOCATED/)
6     B=B+H
      FB=F(B)
      IF((FA.GT.0.0).AND.(FB.LT.0.0)) GO TO 9
      IF((FA.LT.0.0).AND.(FB.GT.0.0)) GO TO 9
      IF(ABS(FA).LT.ABS(FB)) GO TO 8
C*    IF(DABS(FA).LT.DABS(FB)) GO TO 8
      H=0.20E 01*H
      A=B
      FA=FB
      GO TO 5
8     H=-0.20E 01*H
      GO TO 5
C  APPLY BRENT'S ALGORITHM
9     TL=0.200E 01*RF2MQ(0.0)
C*    TL=0.200D 01*DF2MQ(0.0D 00)
      H=0.250E 01*TL
C*    H=0.750D 01*TL
10    C=A
      FC=FA
      D=B-A
      E=D
20    IF(ABS(FC) .GE. ABS(FB)) GO TO 30
```

RF1RF Cont.


```
C*    IF(DABS(FC).GE.DABS(FB)) GO TO 30
      A=B
      B=C
      C=A
      FA=FB
      FB=FC
      FC=FA
30    TL2=TL*ABS(B)+H
C* 30 TL2=TL*DABS(B)+H
      RM=0.500E 00*(C-B)
      IF((ABS(RM) .LE. TL2) .OR. (FB .EQ. 0.0)) GO TO 120
C*    IF((DABS(RM).LE.TL2).OR.(FB.EQ.0.00D 00)) GO TO 120
      IF((ABS(E).LT.TL2) .OR. (ABS(FA).LE.ABS(FB))) GO TO 70
C*    IF((DABS(E).LT.TL2).OR.(DABS(FA).LE.DABS(FB))) GO TO 70
      S=FB/FA
      IF(A.NE.C) GO TO 40
      P=0.200E 01*RM*S
      Q=0.1000E 01-S
      GO TO 50
40    Q=FA/FC
      R=FB/FC
      P=S*(0.200E 01*RM*Q*(Q-R)-(B-A)*(R-0.100E 01))
      Q=(Q-0.10E 01)*(R-0.100E 01)*(S-0.100E 01)
50    IF(P .GT. 0.0) Q=-Q
      IF(P .LE. 0.0) P=-P
      S=E
      E=D
      Q2=0.5000E 00*S*Q
      Q3=0.300E 01*RM*Q-ABS(TL2*Q)
C*    Q3=0.300D 01*RM*Q-DABS(TL2*Q)
      IF(((0.200E 01*P).LT.Q3).AND.(P.LT.ABS(Q2))) GO TO 60
C*    IF(((0.200D 01*P).LT.Q3).AND.(P.LT.DABS(Q2))) GO TO 60
      D=RM
      E=RM
      GO TO 80
60    D=P/Q
      GO TO 80
70    D=RM
      E=RM
80    A=B
      FA=FB
      IF(ABS(D).LE.TL2) GO TO 90
C*    IF(DABS(D).LE.TL2) GO TO 90
      B=B+D
      GO TO 110
```

RF1RF Cont.


```
 90     IF(RM .LE. 0.0E 00) GO TO 100
        B=B+TL2
        GO TO 110
100     B=B-TL2
110     FB=F(B)
        IF((FB .GT. 0.0E 00).AND.(FC .GT. 0.0E 00)) GO TO 10
        IF((FB .LE. 0.0E 00).AND.(FC .LE. 0.0E 00)) GO TO 10
        GO TO 20
120     ROOT=B
        IFAIL=0
        RETURN
        END
```

SUBROUTINE RF1ZP


```
      SUBROUTINE RF1ZP(A,REZ,AIMZ,B,C,NP1,IFAIL)
C  THIS SUBROUTINE COMPUTES THE ROOT OF A REAL POLYNOMIAL
C  ON ENTRY, THE COEFFICIENTS OF THE POLYNOMIAL ARE HELD IN ARRAY A
C  WITH A(1) HOLDING THE COEFFICIENT OF THE HIGHEST POWER
C.    DOUBLE PRECISION A,REZ,AIMZ,B,C,R,RJ,RX,RJX,TL,TL2
C.   *SIG,T,SCALE,G,S1,S2,S,X,Y,FUN,RFUN,FAC,DF2MQ
      LOGICAL SAT
      DIMENSION A(NP1),REZ(NP1),AIMZ(NP1),B(NP1),C(NP1)
      COMMON/RF2ZP/R,RJ,RX,RJX,TL,TL2,SIG,T,SCALE,G,S1,S2,S,X,Y,
     * FUN,RFUN,FAC,N,I,J,K,L,IND2,IC
      IOUT=IF1MQ(1)
      TL=RF2MQ(0.0)
C*    TL=0.20D 01*DF2MQ(0.0)
      N=NP1
      IFAIL=0
      IND2=0
      IC=0
      TL2=TL*SQRT(TL)
C*    TL2=TL*DSQRT(TL)
C  TEST FOR CORRECT DATA
      IF(A(1).NE.0.0 .AND. N.GT.1) GO TO 20
      IFAIL=1
      WRITE(IOUT,100) IFAIL
100   FORMAT(I4,2X,39HFIRST COEFF IS 0.0 OR NUM OF COEFF LT 2/)
      RETURN
20    IF(A(N).NE.0.0) GO TO 30
      N=N-1
      REZ(N)=0.0
      AIMZ(N)=0.0
      IF(N.EQ.1) RETURN
      GO TO 20
30    FAC=0.100E 01
C  PERFORM NORMALISATION,SCHUR TEST AND SEARCH
35    CALL RF3ZP(A,B,C,REZ,AIMZ,NP1)
      IF(N.GT.3) GO TO 40
      IF(N.EQ.1) RETURN
C  OBTAIN LAST ONE OR TWO ROOTS
      CALL RF5ZP(A,REZ,AIMZ,NP1)
      RETURN
C  TEST FOR SADDLE POINT
40    CALL RF4ZP(A,NP1)
      IF((IND2.EQ.2).AND.(IC.GT.3)) GO TO 80
      IF(IND2.EQ.2) GO TO 30
C  EVALUATE POLYNOMIAL AT COMPLEX POINT AND PERFORM ADAMS TEST
50    CALL RF6ZP(A,NP1,SAT)
```

RF1ZP AND AUX.


```
      IF(SAT) GO TO 70
      RFUN=R*R+RJ*RJ
      IF((FUN-RFUN).GE.(SIG*FUN)) GO TO 60
      S1=0.5*S1
      S2=0.5*S2
      S=0.5*S
      SIG=0.5*SIG
      X=X-S1
      Y=Y-S2
      GO TO 50
60    FUN=RFUN
      GO TO 40
C  TEST FOR REAL OR COMPLEX ROOTS AND PERFORM DEFLATION
70    CALL RF7ZP(A,REZ,AIMZ,B,C,NP1,SAT)
      IND2=0
      IC=0
      GO TO 35
80    J=NP1-1
      K=N-1
      NP1=NP1-N
      IFAIL=2
      IF(NP1.EQ.0) RETURN
      WRITE(IOUT,200) IFAIL
      WRITE(IOUT,300) NP1
200   FORMAT(I4,2X,31HCANNOT ESCAPE FROM SADDLE POINT/)
300   FORMAT(6X,4HONLY,I4,1X,14HROOTS COMPUTED/)
      L=0
C  RE-ARRANGE COMPUTED ROOTS BEFORE SADDLE POINT WAS DETERMINED
      DO 90 I=K,J
      L=L+1
      REZ(L)=REZ(I)
      AIMZ(L)=AIMZ(I)
90    CONTINUE
      RETURN
      END

      SUBROUTINE RF3ZP(A,B,C,REZ,AIMZ,NP1)
C.    DOUBLE PRECISION A,B,C,REZ,AIMZ,R,RJ,RX,RJX,TL,TL2,
C.   *SIG,T,SCALE,G,S1,S2,S,X,Y,FUN,RFUN,FAC
      DIMENSION A(NP1),B(NP1),C(NP1),REZ(NP1),AIMZ(NP1)
      COMMON/RF2ZP/R,RJ,RX,RJX,TL,TL2,SIG,T,SCALE,G,S1,S2,S,X,Y,
     * FUN,RFUN,FAC,N,I,J,K,L,IND2,IC
C
C  NORMALISATION
C
```

RF1ZP AND AUX. Cont.

PAGE 3

```
      SCALE=0.0
      DO 10 I=1,N
      IF(ABS(A(I)) .GT. 0.1000E-04) SCALE=SCALE+ALOG(ABS(A(I)))
C*    IF(DABS(A(I)) .GT. 0.100D-06) SCALE=SCALE+DLOG(DABS(A(I)))
10    CONTINUE
      SCALE=SCALE/(FLOAT(N)*0.693147E 00)
      I=INT(SCALE)
C*    I=INT(SNGL(SCALE))
      SCALE=1.0/2.0**I
      DO 20 I=1,N
      B(I)=A(I)*SCALE
20    A(I)=B(I)
      IF(N.LE.3) RETURN
C
C  SCHUR TEST
C
25    DO 50 I=2,N
      J=N-I+2
      DO 30 K=2,J
      L=J-K+1
30    C(K-1)=B(J)*B(K)-B(1)*B(L)
      L=J-1
      IF(C(L) .LT. -TL) GO TO 70
      T=1.0
      IF(C(L) .GT. 1.0) T=1.0/C(L)
      DO 40 K=1,L
40    B(K)=T*C(K)
50    CONTINUE
C
C   TRANSFORMATION
C
      FAC=2.0*FAC
      SCALE=1.0
      L=N-1
      DO 60 I=1,L
      J=L-I+1
      SCALE=2.0*SCALE
      B(J)=A(J)*SCALE
60    A(J)=B(J)
      GO TO 25
C
C     SEARCH
C
70    IF(IND2.NE.2) GO TO 80
      X=0.30E 00+FLOAT(IC)*0.10E 00
```

RF1ZP AND AUX. Cont.

PAGE 4

```
      Y=X+0.10E 00
      IND2=0
      GO TO 90
80    X=0.100000E-02
      Y=0.1
90    CALL RF6ZP(A,NP1,SAT)
      FUN=R*R+RJ*RJ
      RETURN
      END

      SUBROUTINE RF4ZP(A,NP1)
C  CHECK FOR SADDLE POINT
C  ALSO DETERMINE VALID DIRECTION OF SEARCH
C.    DOUBLE PRECISION A,R,RJ,RX,RJX,TL,TL2,
C.   *SIG,T,SCALE,G,S1,S2,S,X,Y,FUN,RFUN,FAC
      DIMENSION A(NP1)
      COMMON/RF2ZP/R,RJ,RX,RJX,TL,TL2,SIG,T,SCALE,G,S1,S2,S,X,Y,
     * FUN,RFUN,FAC,N,I,J,K,L,IND2,IC
      G=RX*RX+RJX*RJX
      IF(G.GE.TL2*FUN) GO TO 60
      IC=IC+1
      IND2=2
      SCALE=1.0
      L=N-1
      DO 50 I=1,L
      J=L-I+1
      SCALE=SCALE*FAC
50    A(J)=A(J)/SCALE
      RETURN
60    S1=-(R*RX+RJ*RJX)/G
      S2=(R*RJX-RJ*RX)/G
      SIG=0.2000E-03
      S=SQRT(S1*S1+S2*S2)
C*    S=DSQRT(S1*S1+S2*S2)
      IF(S.LE.1.0) GO TO 70
      S1=S1/S
      S2=S2/S
      SIG=SIG/S
70    X=X+S1
      Y=Y+S2
      RETURN
      END

      SUBROUTINE RF5ZP(A,REZ,AIMZ,NP1)
C  OBTAIN LAST ONE OR TWO ROOTS
```

RF1ZP AND AUX. Cont.

PAGE 5

```
C.    DOUBLE PRECISION A,REZ,AIMZ,R,RJ,RX,RJX,TL,TL2,
C.   *SIG,T,SCALE,G,S1,S2,S,X,Y,FUN,RFUN,FAC
      DIMENSION A(NP1),REZ(NP1),AIMZ(NP1)
      COMMON/RF2ZP/R,RJ,RX,RJX,TL,TL2,SIG,T,SCALE,G,S1,S2,S,X,Y,
     * FUN,RFUN,FAC,N,I,J,K,L,IND2,IC
      IF(N .EQ. 3) GO TO 10
C  LINEAR
      REZ(1)=-(A(2)/A(1))*FAC
      AIMZ(1)=0.0
      RETURN
C  QUADRATIC
10    R=A(2)*A(2)-0.4000E 01*A(1)*A(3)
      T=SQRT(ABS(R))
C*    T=DSQRT(DABS(R))
      IF(R .GT. 0.0) GO TO 20
      REZ(1)=-0.5*A(2)/A(1)*FAC
      REZ(2)=REZ(1)
      AIMZ(1)=0.5*T/A(1)*FAC
      AIMZ(2)=-AIMZ(1)
      RETURN
20    AIMZ(1)=0.0
      AIMZ(2)=0.0
      REZ(1)=0.5*(-A(2)-SIGN(T,A(2)))/A(1)*FAC
C*    REZ(2)=0.500D 00*(-A(2)-DSIGN(T,A(2)))/A(1)*FAC
      REZ(2)=A(3)/(REZ(1)*A(1))*FAC*FAC
      RETURN
      END

      SUBROUTINE RF6ZP(A,NP1,SAT)
C  EVALUATE POLYNOMIAL AT COMPLEX POINT AND PERFORM ADAMS TEST
C.    DOUBLE PRECISION A,R,RJ,RX,RJX,TL,TL2,
C.   *SIG,T,SCALE,G,S1,S2,S,X,Y,FUN,RFUN,FAC,
C.   *P,Q,A1,A2,A3,B1,B2,B3,C
      LOGICAL SAT
      DIMENSION A(NP1)
      COMMON/RF2ZP/R,RJ,RX,RJX,TL,TL2,SIG,T,SCALE,G,S1,S2,S,X,Y,
     * FUN,RFUN,FAC,N,I,J,K,L,IND2,IC
      P=-2.0*X
      Q=X*X+Y*Y
      T=SQRT(Q)
C*    T=DSQRT(Q)
      A2=0.0
      B2=0.0
      B1=A(1)
      A1=B1
```

RF1ZP AND AUX. Cont.

PAGE 6

```
      C=0.8*ABS(A1)
C*    C=0.8*DABS(A1)
      L=N-2
      DO 10 K=2,L
      A3=A2
      A2=A1
      A1=A(K)-P*A2-Q*A3
      C=T*C+ABS(A1)
C*    C=T*C+DABS(A1)
      B3=B2
      B2=B1
10    B1=A1-P*B2-Q*B3
      A3=A2
      A2=A1
      A1=A(N-1)-P*A2-Q*A3
      R=A(N)+X*A1-Q*A2
      RJ=A1*Y
      RX=A1-2.0*B2*Y*Y
      RJX=2.0*Y*(B1-X*B2)
      C=T*(T*C+ABS(A1))+ABS(R)
C*    C=T*(T*C+DABS(A1))+DABS(R)
      P=SQRT(R*R+RJ*RJ)
C*    P=DSQRT(R*R+RJ*RJ)
      Q=2.0*ABS(X*A1)-8.0*(ABS(R)+ABS(A1)*T)
C*    Q=0.200D 01*DABS(X*A1)-0.800D 01*(DABS(R)+DABS(A1)*T)
      Q=(Q+10.0*C)*TL*(1.0+TL)**(4*N-1)
      SAT=.FALSE.
      IF(P .LT. Q) SAT=.TRUE.
      RETURN
      END

      SUBROUTINE RF7ZP(A,REZ,AIMZ,B,C,NP1,SAT)
C  THIS ROUTINE CHECKS FOR REAL AND COMPLEX ROOTS AND PERFORM
C  COMPOSITE DEFLATION
C.    DOUBLE PRECISION A,REZ,AIMZ,B,C,R,RJ,RX,RJX,TL,TL2,
C.   *SIG,T,SCALE,G,S1,S2,S,X,Y,FUN,RFUN,FAC
      LOGICAL SAT
      DIMENSION A(NP1),REZ(NP1),AIMZ(NP1),B(NP1),C(NP1)
      COMMON/RF2ZP/R,RJ,RX,RJX,TL,TL2,SIG,T,SCALE,G,S1,S2,S,X,Y,
     * FUN,RFUN,FAC,N,I,J,K,L,IND2,IC
      FUN=1.0/TL2
      K=1
      AIMZ(N-1)=Y*FAC
      IF(ABS(Y) .GT. 1.0) GO TO 20
C*    IF(DABS(Y) .GT. 0.100D 01) GO TO 20
```

RF1ZP AND AUX. Cont.


```
C  CHECK FOR REAL ROOT
      S1=Y
      Y=0.0
      CALL RF6ZP(A,NP1,SAT)
      Y=S1
      IF(.NOT.SAT) GO TO 20
C  REAL ROOT ACCEPTED AND BOTH BACKWARD AND FORWARD DEFLATION ARE
C  PERFORMED WITH LINEAR FACTOR
      REZ(N-1)=X*FAC
      AIMZ(N-1)=0.0
      N=N-1
      B(1)=A(1)
      C(N)=-A(N+1)/X
      DO 10 I=2,N
      B(I)=A(I)+X*B(I-1)
      L=N-I
10    C(L+1)=(C(L+2)-A(L+2))/X
      GO TO 40
C
C   COMPLEX
C
20    REZ(N-1)=X*FAC
      REZ(N-2)=REZ(N-1)
      AIMZ(N-2)=-AIMZ(N-1)
      N=N-2
      R=X+X
      RJ=-(X*X+Y*Y)
      B(1)=A(1)
      B(2)=A(2)+R*B(1)
      C(N)=-A(N+2)/RJ
      C(N-1)=-(A(N+1)+R*C(N))/RJ
      DO 30 I=3,N
      B(I)=A(I)+R*B(I-1)+RJ*B(I-2)
      L=N-I
30    C(L+1)=-(A(L+3)-C(L+3)+R*C(L+2))/RJ
C
C   J O I N
C
40    DO 50 I=1,N
      RFUN=ABS(B(I))+ABS(C(I))
C*    RFUN=DABS(B(I))+DABS(C(I))
      IF(RFUN .LE. TL) GO TO 50
      RFUN=ABS(B(I)-C(I))/RFUN
C*    RFUN=DABS(B(I)-C(I))/RFUN
      IF(RFUN .GE. FUN) GO TO 50
```

RF1ZP AND AUX. Cont.

PAGE 8

```
316        FUN=RFUN
317        K=I
318 50     CONTINUE
319        IF(K .LT. 2) GO TO 70
320        L=K-1
321        DO 60 I=1,L
322 60     A(I)=B(I)
323 70     A(K)=0.500E 00*(B(K)+C(K))
324        L=K+1
325        IF(L.GT.N) RETURN
326        DO 80 I=L,N
327 80     A(I)=C(I)
328        RETURN
329        END
```

APPENDIX D

SUBROUTINES FOR QUADRATURE

The subroutines in this appendix approximate the definite integral:

$$\int_a^b f(x)dx$$

and table D summarizes the action of the subroutines.

SUBROUTINE NAME	PURPOSE
RF1QU	computes an approximation to the definite integral $\int_a^b f(x)dx$ where a and b are finite and f(x) is defined in (a,b)
RF4QU	Computes an approximation to the definite integral $\int_{x_0}^{\infty} e^{-x}f(x)\,dx$ where x_0 is finite
RF5QU	Perform the numerical integration of unequally spaced data

TABLE D: A summary of subroutines for integration

1. SUBROUTINE RF1QU

a) Purpose

The subroutine RF1QU integrates a given function over a given finite internal. The function need not be defined at the end-points.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

SUBROUTINE RF1QU(A,B,F,EPS, ANS, RELER, IFAIL)

c) Description

This subroutine approximates the integral

$$\int_a^b f(x)dx$$

by using a family of interlacing whole-interval common-point, quadrature formulas in an adaptive manner. This family is called Patterson's family and RF1QU is a modified version of Patterson's subroutine for adaptive integration (see Patterson (131) for more details). The relative error used for computing the solution is the sum of the user supplied relative error and the machine relative error. This reduces the danger of a user requesting for an accuracy which cannot be attained.

d) Parameters

A real variable, which on entry, the user must set to the lower limit of integration. It is not altered on exit.

B real variable which on entry, the user must set to the upper limit of integration. It is unaltered on exit.

F real function, supplied by the user. It defines the function whose integral over a finite interval is required. It's specification is

```
REAL FUNCTION F(X)
REAL  X
```

F must be declared as EXTERNAL in the (sub) program from which RF1QU is called.

EPS real variable on entry, it specifies the relative accuracy to which the integral is required. It is unaltered on exit. EPS should be chosen to be smaller than 0.1E-02 since the reliability of the subroutine is reduced for large EPS.

ANS real variable, on exit, ANS gives an approximation to the integral.

RELER real variable, on exit, RELER gives the value of the estimated error in the computed integral.

IFAIL integer variable, and it is set to zero on successful exit from RF1QU, otherwise it is set to a non-zero value.

NOTE: All real variables and functions must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is unaltered by the subroutine can be replaced with a constant.

e) Error diagnostics:

A successful return from RF1QU is indicated by IFAIL equal

zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	CONVERGENCE NOT ATTAINED WITHIN LIMITATION OF ROUTINE	Increase EPS, if problem persists, then the integrand is badly behaved (possibly at the end points). Check RELER. A result is normally obtained but it may be unreliable because it was obtained by ignoring convergence failures in some sub-intervals when the internally declared stack was full or sub-intervals were too small.
2	A RELAXED CONVERGENCE CRITERION WAS USED	Converged result was obtained but at some point a relaxed convergence criterion was invoked. EPS and RELER may be checked for any discrepancy.

f) Other routines used:

IF1QU calls subroutine RF2QU. RF2QU is not used by any other routine in library. It is actually a part of RF1QU. FUNCTIONS IF1MQ and RF2MQ are referenced by RF1QU. The user is not aware of these routines.

g) Timing

The amount of time taken depends on the function being integrated. For the function given in the example, it took 0.08 sec to compute its integral, using Tx990/4 microcomputer.

h) Storage

There are internally declared arrays and the compiled output of RF1QU together with RF2QU using Tx990/4 microcomputer occupies 2,810 bytes. There is a labelled common storage named RF3QU.

i) Accuracy

The result produced is accurate to the requested accuracy or to machine relative error if IFAIL is zero on exit.

j) Example

RF1QU was used to integrate the function

$$f(x) = x/(e^{x}-1.0) \text{ over the range } (0,1)$$

Example D1 shows how RF1QU can be used to integrate f(x). The requested accuracy was 0.1E-05 and the solution is given as 0.775035E00 with an estimated relative error of 0.61326E-06. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

2. SUBROUTINE RF4QU

a) Purpose

RF4QU integrates functions of the type $e^{-x}f(x)$ over a semi-infinite interval.

b) Specification

The subroutine is written in FORTRAN and it is of the form

```
SUBROUTINE RF4QU (XO, ANS, FUN)
```

c) Description

RF4QU approximates the semi-infinite integral

$$\int_{x_0}^{\infty} e^{-x}f(x)dx$$

by using the laguerre formula given by

EXAMPLE D1

```
0001        REAL ANS,RELER
0002        EXTERNAL FUN
0003        DATA INPUT,IOUT/1,2/
0004        CALL RF1QU(0.0,1.0,FUN,0.10E-05,ANS,RELER,IFAIL)
0005        WRITE(IOUT,100) ANS,RELER,IFAIL
0006 100    FORMAT(4X,E14.7,2X,E14.7,2X,I4)
0007        STOP
0008        END
0009        REAL FUNCTION FUN(X)
0010        FUN=X/(EXP(X)-0.1E 01)
0011        RETURN
0012        END
0013 RESULT
0014        0.7775035E 00   0.6132926E-06     0
```

$$\int_0^{\infty} e^{-x} f(x)dx = \sum_{k=1}^{n} w_k f(x_k)$$

where the abscissas x_k are the zeros of the laguerre polynomial Ln(x) ($Ln(x) = (-1)^n x^n + \ldots$)

and

$$w_k = \frac{(n!)^2 x_k}{(Ln+1 (x_k))^2}$$

The values of w_k and x_k can be obtained from tables for a given n (see Stroud and Secrest (157)). For the single precision implementation, n is chosen as sixteen while for double precision implementation n is chosen as twenty-five. During the summation $w_k f(x_k)$, a check is carried out to test whether there is any improvement in the sum. If there is none, the summation is terminated.

d) <u>Parameters</u>

XO — real variable and on entry, XO specifies the lower bound of integration. Unaltered on exit.

ANS — real variable and on exit, ANS gives an approximation to the integral.

FUN — real FUNCTION, supplied by the user. It defines the function whose integral over a semi-infinite interval is to be determined. It's specification is:

```
REAL FUNCTION FUN(x)
REAL X
```

FUN must be declared as EXTERNAL in (sub) program from which RF4QU is called. As an example, if the integral is:

$$\int_{x_0}^{\infty} xe^{-x}\, dx,$$

then FUN(x) = x and if the integral is

$$\int_{x_0}^{\infty} \mathrm{Sin}(x)\, dx$$

then FUN (x) = $e^{x} \sin(x)$

NOTE: All real variables and functions must be declared as DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is unaltered by the subroutine can be replaced by a constant.

e) Error diagnostics

None

f) Other routines used

The FUNCTION RF2MQ of the library is referenced.

g) Timing

Depends on the function and for the example given using Tx990/4 microcomputer, the time taken was 0.06 sec.

h) Storage

There are internally declared arrays and the compiled output of RF4QU using Tx990/4 microcomputer is 432 bytes.

i) Accuracy

Depends on the behaviour of the function. If the integral is of the form

$$\int_{x_0}^{\infty} e^{-x} f(x)$$

then high accuracy is expected, but integrals of other forms of functions are less accurate.

j) Example

Example D2 shows how the subroutine can be used. This single

precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

In this example, the integral

$$\int_{1}^{\infty} xe^{-x}dx$$

is evaluated. The result obtained is given to be 0.7357571E 0

k) Further Comments

RF4QU can be used to approximate the integral

$$\int_{-\infty}^{\infty} g(x)dx$$

by splitting it into

$$\int_{0}^{\infty} g(x)dx + \int_{-\infty}^{0} g(x)dx = \int_{0}^{\infty} g(x)dx + \int_{0}^{\infty} g(-x)dx$$

3. SUBROUTINE RF5QU

a) Purpose

RF5QU performs numerical integration of unequally spaced data.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

SUBROUTINE RF5QU (X, Y, RINT4, N, IA, IB, E, IFAIL)

c) Description

RF5QU is based on the classical integration formula that uses divided differences. In addition, an indication of the reliability of the result obtained is tested by comparing it with the corresponding result obtained by a process of piecewise

EXAMPLE D2

```
0001        REAL ANS
0002        EXTERNAL FUN
0003        DATA INPUT,IOUT/1,2/
0004        CALL RF4QU(1.0,ANS,FUN)
0005        WRITE(IOUT,100) ANS
0006 100    FORMAT(4X,E14.7/)
0007        STOP
0008        END
0009        REAL FUNCTION FUN(X)
0010        FUN=X
0011        RETURN
0012        END
0013 RESULT
0014      0.7357571E 00
```

quartic interpolation. Although, the degree of consistency observed does not guarantee a like degree of "accuracy" in any absolute sense, in a great majority of practical applications, the consistency check may be expected to furnish a good indication of the extent to which the integral is reliably defined, by the data. (see Gill and Miller (72)).

d) Parameters

X	real one-dimensional array of size N. On entry, the array X specifies the abscissas. Two values must not be the same and it is preferable for the values to be arranged in ascending order.
Y	real one-dimensional array of size N. On entry the array Y specifies the corresponding ordinates to X.
RINT4	real variable, on exit RINT4 contains an approximation of the integral.
N	integer variable, on entry N specifies the number of co-ordinates. N is unaltered on exit.
IA	integer variable, on entry IA specifies the lower subscript of array X from which the integration is to start. IA is unaltered.
IB	integer variable, on entry, IB specifies the upper subscript of array X which is the limit of integration. IB must not be greater than N. IB is unaltered on exit.

E real variable, on exit E gives the estimated error in the computed integral.

IFAIL integer variable, it is set to zero on successful exit from RF5QU, otherwise it is set to a non zero value.

NOTE: All real variables and functions must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

A successful return from RF5QU is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	NUMBER OF PTS LT 4 OR LOWER SUBSCRIPT GT UPPER ONE	Increase number of points or check IA and IB and take necessary steps
2	LOWER SUBSCRIPT LT 1 OR UPPER SUBSCRIPT GT NUM OF GIVEN VALUES	Check whether IA is less than one or IB is greater than N. Take necessary action.
3	SOME VALUES OF X CO-ORDINATE ARE REPEATED	Check the values of array X for equal values and make the necessary change.

f) Other routines used

Function IF1MQ is referenced by RF5QU

g) Timing

Depends on the number of data points and for the given example it took 0.2 sec to compute the integral using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF5QU using Tx990/4 microcomputer occupies 2,266 bytes.

i) Accuracy

Depends on the data points given. However an estimate of the error in the given result is calculated.

j) Example

RF5QU was used to perform the numerical integration of a set of data points given in Example D3. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

4. Listing of subroutines

EXAMPLE D3

```
0001        REAL X,Y,RINT4,E
0002        INTEGER IA,IB,N,IFAIL
0003        DIMENSION X(20),Y(20)
0004        DATA INPUT,IOUT/1,2/
0005        READ(INPUT,100) N
0006        READ(INPUT,200)(X(I),Y(I),I=1,N)
0007        IA=1
0008        IB=N
0009        CALL RF5QU(X,Y,RINT4,N,IA,IB,E,IFAIL)
0010        WRITE(IOUT,300) RINT4,E,IFAIL
0011 100    FORMAT(I4)
0012 200    FORMAT(2E14.7)
0013 300    FORMAT(E14.7,3X,E14.7,3X,I4)
0014        STOP
0015        END
0016 DATA
0017   10
0018  0.5000000E-01 0.2236068E 00
0019  0.1000000E 00 0.3162278E 00
0020  0.2500000E 00 0.5000000E 00
0021  0.3000000E 00 0.5477226E 00
0022  0.4500000E 00 0.6708204E 00
0023  0.5000000E 00 0.7071068E 00
0024  0.6000000E 00 0.7745967E 00
0025  0.7000000E 00 0.8366600E 00
0026  0.8000000E 00 0.8944272E 00
0027  0.1000000E 00 0.1000000E 00
0028 RESULT
0029  0.6594087E 00    [illegible].1073904E-03      0
```

SUBROUTINE RF1QU

PAGE 1

```
      SUBROUTINE RF1QU(A,B,F,EPS,ANS,RELER,IFAIL)
C  THIS ROUTINE PERFORMS AUTOMATIC INTEGRATION OVER A FINITE
C  INTERVAL USING THE BASIC INTEGRATION ALGORITHM RF2QU
C.    DOUBLE PRECISION A,B,ANS,RELER,F,RST,STACK,SUB1,
C.   *   SUB2,SUB3,ESTIM,COMP,TL,TL2,EPS,DF2MQ
      DIMENSION RST(5),STACK(150)
      DATA ISMAX/150/
      IOUT=IF1MQ(1)
      TL=2.0*RF2MQ(0.0)
C*    TL=100.0*DF2MQ(0.0)
      TL2=EPS+TL
      CALL RF2QU(A,B,RST,K,TL2,IFAIL,F)
      ANS=RST(K)
      RELER=0.0E 00
      IF(ANS.NE.0.0) RELER=ABS((RST(K)-RST(K-1))/ANS)
C*    IF(ANS.NE.0.0) RELER=DABS((RST(K)-RST(K-1))/ANS)
C  CHECK IF SUBDIVISION IS NEEDED
      IF(IFAIL.EQ.0) RETURN
C  SUBDIVIDE
      ESTIM=ABS(ANS*TL)
C*    ESTIM=DABS(ANS*TL)
      RELER=0.0
      ANS=0.0
      IS=1
      IC=1
      SUB1=A
      SUB3=B
10    SUB2=0.5*(SUB1+SUB3)
      CALL RF2QU(SUB1,SUB2,RST,K,TL2,IFAIL,F)
      COMP=ABS(RST(K)-RST(K-1))
C*    COMP=DABS(RST(K)-RST(K-1))
      IF(IFAIL.EQ.0) GO TO 30
      IF(COMP.LE.ESTIM) GO TO 70
      IF(ABS(SUB2-SUB1).LE.TL) GO TO 20
C*    IF(DABS(SUB2-SUB1).LE.TL) GO TO 20
      IF(IS.GE.ISMAX) GO TO 20
C  STACK SUBINTERVAL (SUB1,SUB2) FOR FUTURE EXAMINATION
      STACK(IS)=SUB1
      IS=IS+1
      STACK(IS)=SUB2
      IS=IS+1
      GO TO 40
20    IC=-IABS(IC)
30    ANS=ANS+RST(K)
      RELER=RELER+COMP
40    CALL RF2QU(SUB2,SUB3,RST,K,TL2,IFAIL,F)
      COMP=ABS(RST(K)-RST(K-1))
C*    COMP=DABS(RST(K)-RST(K-1))
      IF(IFAIL.EQ.0) GO TO 50
      IF(COMP.LE.ESTIM) GO TO 80
```

RF1QU AND AUX.


```
      IF(ABS(SUB3-SUB2).GT.TL) GO TO 45
C*    IF(DABS(SUB3-SUB2).GT.TL) GO TO 45
      IC=-IABS(IC)
      GO TO 50
C  SUBDIVIDE INTERVAL (SUB2,SUB3)
45    SUB1=SUB2
      GO TO 10
50    ANS=ANS+RST(K)
      RELER=RELER+COMP
      IF(IS.EQ.1) GO TO 60
C  SUBDIVIDE THE DELIQUENT INTERVAL LAST STACKED
      IS=IS-1
      SUB3=STACK(IS)
      IS=IS-1
      SUB1=STACK(IS)
      GO TO 10
C   RELAXED CONVERGENCE
70    IC=ISIGN(2,IC)
      GO TO 30
80    IC=ISIGN(2,IC)
      GO TO 50
C  SUBDIVISION RESULT
60    IFAIL=0
      IF(IC.LT.0) IFAIL=1
      IF(IC.EQ.2) IFAIL=2
      IF(IFAIL.EQ.1) WRITE(IOUT,100) IFAIL
      IF(IFAIL.EQ.2) WRITE(IOUT,90) IFAIL
      RELER=RELER/ABS(ANS)
C*    RELER=RELER/DABS(ANS)
90    FORMAT(2X,I2,3X,39HA RELAXED CONVERGENCE CRITERIA WAS USED/)
100   FORMAT(2X,I2,3X,24HCONVERGENCE NOT ATTAINED/
     1,7X,28HWITHIN LIMITATION OF ROUTINE/)
      RETURN
      END


      SUBROUTINE RF2QU(A,B,RST,K,TL,ICHEK,F)
C
C  THIS ROUTINE ATTEMPTS TO CALCULATE THE INTEGRAL OF F(X)
C  FOR THE INTERVAL (A,B) TO AN ACCURACY OF AT LEAST TL
C
C.    DOUBLE PRECISION A,B,RST,TL,F,PQ,FUNCT,DIFF,X,SUM,FZERO
C-    DOUBLE PRECISION ACC
      DIMENSION FUNCT(15),RST(5),PQ(48)
      COMMON/RF3QU/PQ
C
      ICHEK=0
C     CHECK FOR TRIVIAL CASE
      IF(A.NE.B) GO TO 1
      K=2
```

RF1QU AND AUX. Cont.


```
      RST(1)=0.0
      RST(2)=0.0
      NPTS=0
      RETURN
C  SCALE FACTORS
1     SUM=0.500E 00*(B+A)
      DIFF=0.5000E 00*(B-A)
C  1-POINT GAUSS
      FZERO=F(SUM)
      RST(1)=0.200E 01*FZERO*DIFF
      I=0
      IOLD=0
      NEW=1
      K=2
      ACC=0.0
      GO TO 30
10    IF(K.EQ.5) GO TO 50
      K=K+1
      ACC=0.0
C  CONTRIBUTION FROM FUNCTION VALUES ALREADY COMPUTED
      DO 20 J=1,IOLD
      I=I+1
      ACC=ACC+PQ(I)*FUNCT(J)
C+    ACC=ACC+PQ(I)*DBLE(FUNCT(J))
20    CONTINUE
C  CONTRIBUTION FROM NEW FUNCTION VALUES.
30    IOLD=IOLD+NEW
      DO 40 J=NEW,IOLD
      I=I+1
      X=PQ(I)*DIFF
      FUNCT(J)=F(SUM+X)+F(SUM-X)
      I=I+1
      ACC=ACC+PQ(I)*FUNCT(J)
C+    ACC=ACC+PQ(I)*DBLE(FUNCT(J))
40    CONTINUE
      NEW=IOLD+1
      I=I+1
      RST(K)=(ACC+PQ(I)*FZERO)*DIFF
C+    RST(K)=SNGL((ACC+PQ(I)*DBLE(FZERO))*DIFF)
C  CHECK FOR CONVERGENCE
      IF(ABS(RST(K)-RST(K-1))-TL*ABS(RST(K))) 60,60,10
C*    IF(DABS(RST(K)-RST(K-1))-TL*DABS(RST(K))) 60,60,10
C  CONVERGENCE NOT ACHIEVED.
50    ICHEK=1
C  NORMAL TERMINATION.
60    NPTS=NEW+IOLD
      RETURN
      END
```

RF1QU AND AUX. Cont.


```
      BLOCK DATA
C     DOUBLE PRECISION P
      DIMENSION P(48)
      COMMON/RF3QU/P
      DATA P(1),P(2)/
     2 0.7745966E 00,0.5555555E 00/
C*   1 0.7745966692414834D 00,0.5555555555555556D 00/
      DATA P(3),P(4)/
     2 0.8888888E 00,0.2684880E 00/
C*   1 0.8888888888888889D 00,0.2684880898683334D 00/
      DATA P(5),P(6)/
     2 0.9604912E 00,0.1046562E 00/
C*   1 0.9604912687080203D 00,0.1046562260264673D 00/
      DATA P(7),P(8)/
     2 0.4342437E 00,0.4013974E 00/
C*   1 0.4342437493468026D 00,0.4013974147759622D 00/
      DATA P(9),P(10)/
     2 0.4509165E 00,0.1344152E 00/
C*   1 0.4509165386584741D 00,0.1344152552437842D 00/
      DATA P(11),P(12)/
     2 0.5160328E-01,0.2006285E 00/
C*   1 0.5160328299707974D-01,0.2006285293769890D 00/
      DATA P(13),P(14)/
     2 0.9938319E 00,0.1700171E-01/
C*   1 0.9938319632127550D 00,0.1700171962994026D-01/
      DATA P(15),P(16)/
     2 0.8884592E 00,0.9292719E-01/
C*   1 0.8884592326722570D 00,0.9292719531512454D-01/
      DATA P(17),P(18)/
     2 0.6211029E 00,0.1715119E 00/
C*   1 0.6211029467372264D 00,0.1715119091363914D 00/
      DATA P(19),P(20)/
     2 0.2233866E 00,0.2191568E 00/
C*   1 0.2233866864289669D 00,0.2191568584015875D 00/
      DATA P(21),P(22)/
     2 0.2255104E 00,0.6720775E-01/
C*   1 0.2255104997982067D 00,0.6720775429599070D-01/
      DATA P(23),P(24)/
     2 0.2580759E-01,0.1003142E 00/
C*   1 0.2580759809617665D-01,0.1003142786117956D 00/
      DATA P(25),P(26)/
     2 0.8434565E-02,0.4646289E-01/
C*   1 0.8434565739321106D-02,0.4646289326175799D-01/
      DATA P(27),P(28)/
     2 0.8575592E-01,0.1095784E 00/
C*   1 0.8575592004999035D-01,0.1095784210559246D 00/
      DATA P(29),P(30)/
     2 0.9990981E 00,0.2544780E-02/
C*   1 0.9990981249676676D 00,0.2544780791561874D-02/
      DATA P(31),P(32)/
```

RF1QU AND AUX. Cont.


```
      2 0.9815311E 00,0.1644604E-01/
C*    1 0.9815311495537401D 00,0.1644604985438781D-01/
       DATA P(33),P(34)/
      2 0.9296548E 00,0.3595710E-01/
C*    1 0.9296548574297401D 00,0.3595710330712932D-01/
       DATA P(35),P(36)/
      2 0.8367259E 00,0.5697950E-01/
C*    1 0.8367259361688687D 00,0.5697950949412336D-01/
       DATA P(37),P(38)/
      2 0.7024962E 00,0.7687962E-01/
C*    1 0.7024962064915271D 00,0.7687962049900353D-01/
       DATA P(39),P(40)/
      2 0.5313197E 00,0.9362710E-01/
C*    1 0.5313197436443756D 00,0.9362710998126447D-01/
       DATA P(41),P(42)/
      2 0.3311353E 00,0.1056698E 00/
C*    1 0.3311353932579768D 00,0.1056698935802348D 00/
       DATA P(43),P(44)/
      2 0.1124889E 00,0.1119568E 00/
C*    1 0.1124889431331866D 00,0.1119568730209535D 00/
       DATA P(45),P(46)/
      2 0.1127552E 00,0.3360387E-01/
C*    1 0.1127552567207687D 00,0.3360387714820773D-01/
       DATA P(47),P(48)/
      2 0.1290380E-01,0.5015713E-01/
C*    1 0.1290380010095127D-01,0.5015713930589954D-01/
C
       END
```

SUBROUTINE RF4QU

PAGE 1

```
      SUBROUTINE RF4QU(X0,ANS,FUN)
C.    DOUBLE PRECISION A,X,Q,XT,X0,AREA,ANS,FUN,DF2MQ
      DIMENSION A(16),X(16)
C*    DIMENSION A(25),X(25)
      DATA X(1),A(1)/0.87649410E-01,0.20615171E 00/
C*    DATA X(1),A(1)/0.5670477545270547D-01,0.1375260142293443D 00/
      DATA X(2),A(2)/0.46269632E 00,0.33105785E 00/
C*    DATA X(2),A(2)/0.2990108985869885D 00,0.2516452737649102D 00/
      DATA X(3),A(3)/0.11410577E 01,0.26579577E 00/
C*    DATA X(3),A(3)/0.7359095554330161D 00,0.2561760028097558D 00/
      DATA X(4),A(4)/0.21292836E 01,0.13629693E 00/
C*    DATA X(4),A(4)/0.1369183116035193D 01,0.1842154903624364D 00/
      DATA X(5),A(5)/0.34370846E 01,0.47028928E-01/
C*    DATA X(5),A(5)/0.2201326053721467D 01,0.1031998481075206D 00/
      DATA X(6),A(6)/0.50780186E 01,0.11299900E-01/
C*    DATA X(6),A(6)/0.3235675803558037D 01,0.4471416112993353D-01/
      DATA X(7),A(7)/0.70703385E 01,0.18490709E-02/
C*    DATA X(7),A(7)/0.4476496615073834D 01,0.1530523286395430-01/
      DATA X(8),A(8)/0.94383143E 01,0.20427191E-03/
C*    DATA X(8),A(8)/0.5929083762700447D 01,0.4152414632877073D-02/
      DATA X(9),A(9)/0.12214223E 02,0.14844586E-04/
C*    DATA X(9),A(9)/0.7599899309956749D 01,0.8920990732596811D-03/
      DATA X(10),A(10)/0.15441527E 02,0.68283193E-06/
C*    DATA X(10),A(10)/0.9496749220932434D 01,0.1511560191642405D-03/
      DATA X(11),A(11)/0.19180156E 02,0.18810248E-07/
C*    DATA X(11),A(11)/0.1162901491177875D 02,0.2006553180193295D-04/
      DATA X(12),A(12)/0.23515905E 02,0.28623502E-09/
C*    DATA X(12),A(12)/0.1400795797654506D 02,0.2067774396431877D-05/
      DATA X(13),A(13)/0.28578729E 02,0.21270790E-11/
C*    DATA X(13),A(13)/0.1664712559728878D 02,0.1634652022291141D-06/
      DATA X(14),A(14)/0.34583398E 02,0.62979670E-14/
C*    DATA X(14),A(14)/0.1956289801146905D 02,0.9766015062124477D-08/
      DATA X(15),A(15)/0.41940452E 02,0.50504737E-17/
C*    DATA X(15),A(15)/0.2277524198683504D 02,0.4327720794184926D-09/
      DATA X(16),A(16)/0.51701160E 02,0.41614623E-21/
C*    DATA X(16),A(16)/0.2630877239096888D 02,0.1389600963389494D-10/
C.    DATA X(17),A(17)/0.3019429116331610D 02,0.3138922792539925D-12/
C.    DATA X(18),A(18)/0.3447109757192203D 02,0.4802614822604204D-14/
C.    DATA X(19),A(19)/0.3919060880393742D 02,0.4735685364807236D-16/
C.    DATA X(20),A(20)/0.4442234933616201D 02,0.2814205379843026D-18/
C.    DATA X(21),A(21)/0.5026457499383353D 02,0.9164954395991140D-21/
C.    DATA X(22),A(22)/0.5686496717394017D 02,0.1418940009497222D-23/
C.    DATA X(23),A(23)/0.6446667061595413D 02,0.8273651944099207D-27/
C.    DATA X(24),A(24)/0.7353423479210014D 02,0.1168861711542765D-30/
C.    DATA X(25),A(25)/0.8526015556249595D 02,0.1315831500059125D-35/
      AREA=0.0E 00
      TL=0.5E 00*RF2MQ(0.0)
C*    TL=0.5E 00*DF2MQ(0.0)
      DO 10 I=1,16
C*    DO 10 I=1,25
```

RF4QU Cont.


```
      XT=X(I)+X0
      Q=A(I)*FUN(XT)
      AREA=AREA+Q
      IF(ABS(Q/AREA).LE.TL) GO TO 20
C*    IF(DABS(Q/AREA)).LE.TL) GO TO 20
10    CONTINUE
20    ANS=EXP(-X0)*AREA
C* 20 ANS=DEXP(-X0)*AREA
      RETURN
      END
```

SUBROUTINE RF5QU


```
      SUBROUTINE RF5QU(X,Y,RINT4,N,IA,IB,E,IFAIL)
C     DOUBLE PRECISION X,Y,E,H1,H2,H3,H4,R1,R2,R3,R4,
C    1   D1,D2,D3,C,S,RINT,TEMP
      DIMENSION X(N),Y(N)
      IOUT=IF1MB(1)
      IFAIL=1
      E=0.0
      IF((N.LT.4).OR.(IA.GT.IB)) GO TO 1
      GO TO 5
1     WRITE(IOUT,200) IFAIL
      WRITE(IOUT,250)
      RETURN
200   FORMAT(2X,I2,3X,37HNUMBER OF PTS LT 4 OR LOWER SUBSCRIPT/)
250   FORMAT(7X,12HGT UPPER ONE/)
5     IF((IA.LT.1).OR.(IB.GT.N)) GO TO 8
      GO TO 9
8     IFAIL=2
      WRITE(IOUT,400) IFAIL
      WRITE(IOUT,450)
      RETURN
400   FORMAT(2X,I2,3X,39HLOWER SUBSCRIPT LT 1 OR UPPER SUBSCRIPT/)
450   FORMAT(7X,22HGT NUM OF GIVEN POINTS/)
9     IF(IA .NE. IB) GO TO 10
      RINT4=0.0
      IFAIL=0
      RETURN
10    RINT=0.0
      S=0.0
      C=0.0
      R4=0.0
      J=3
      IF((IA.EQ.(N-1)) .AND. N.GT.4) GO TO 20
      IF(IA.GT.3) J=IA
      GO TO 30
20    J=N-2
30    K=N-1
      IF(IB.EQ.2 .AND. N.GT.4) GO TO 40
      IF(N .GT. (IB+2)) K=IB+1
      GO TO 50
40    K=4
50    DO 120 I=J,K
      IF(I.NE.J) GO TO 60
      H2=X(J-1)-X(J-2)
      IF(H2.EQ.0.0E 00) GO TO 130
      D3=(Y(J-1)-Y(J-2))/H2
      H3=X(J)-X(J-1)
      IF(H3.EQ.0.0E 00) GO TO 130
      D1=(Y(J)-Y(J-1))/H3
      H1=H2+H3
      D2=(D1-D3)/H1
```

RF5QU Cont.


```
      H4=X(J+1)-X(J)
      IF(H4.EQ.0.0E 00) GO TO 130
      R1=(Y(J+1)-Y(J))/H4
      R2=(R1-D1)/(H4+H3)
      H1=H1+H4
      R3=(R2-D2)/H1
      IF(IA.NE.1) GO TO 70
      RINT=H2*(Y(1)+H2*(0.500E 00*D3-H2*(D2/0.600E 01-(H2+0.20E 01
     2     *H3)*R3/0.1200E 02)))
      S=-H2*H2*H2*(H2*(0.30E 01*H2+0.5000E 01*H4)
     2    +0.100E 02*H3*H1)/0.6000E 02
      GO TO 70
   60 H4=X(I+1)-X(I)
      IF(H4.EQ.0.0E 00) GO TO 130
      R1=(Y(I+1)-Y(I))/H4
      R4=H4+H3
      R2=(R1-D1)/R4
      R4=R4+H2
      R3=(R2-D2)/R4
      R4=R4+H1
      R4=(R3-D3)/R4
   70 IF((I.LE.IB).AND.(I.GT.IA)) GO TO 80
      E=E+R4*S
      GO TO 90
   80 TEMP=H3*H3
      RINT=RINT+H3*((Y(I)+Y(I-1))*0.500E 00-TEMP*(D2+R2+(H2-H4)
     2     *R3)/0.12000E 02)
      C=TEMP*H3*(0.200E 01*TEMP+0.50000E 01*(H3*(H4+H2)
     2   +0.200E 01*H4*H2))/0.120000E 03
      E=E+(C+S)*R4
      S=0.200E 01*C+S
      IF(I.NE.J) S=C
   90 IF(I.NE.K) GO TO 110
      IF(IB.NE.N) GO TO 100
      RINT=RINT+H4*(Y(N)-H4*(0.500E 00*R1+H4*(R2/0.60E 01
     2     +(0.20E 01*H3+H4)*R3/0.1200E 02)))
      E=E-H4*H4*H4*R4*(H4*(0.300E 01*H4+0.5000E 01*H2)
     2  +0.100E 02*H3*(H2+H3+H4))/0.6000E 02
  100 IF(IB.GE.(N-1)) E=E+S*R4
      GO TO 120
  110 H1=H2
      H2=H3
      H3=H4
      D1=R1
      D2=R2
      D3=R3
  120 CONTINUE
      RINT4=RINT
      IFAIL=0
      RETURN
```

RF5QU Cont.

PAGE 3

```
130   IFAIL=3
      WRITE(IOUT,300) IFAIL
300   FORMAT(2X,I2,3X,41HSOME VALUES OF X CO-ORDINATE ARE REPEATED/)
      RETURN
      END
```

APPENDIX E

SUBROUTINE FOR ORDINARY DIFFERENTIAL EQUATIONS

The subroutines in this appendix solve initial-value problem of the form (a)

$$y_i' = g_i\ (t, y, \ldots, y_n)\ i = 1, 2, \ldots, n$$

with initial conditions $y_i(t_o) = \beta_i$ $i = 1, \ldots, n$
where n is a finite integer and (b) second order boundary-value problems of the form:

$$y_1' = g_1(t, y_1, y_2)$$

$$y_2' = g_2(t, y_1, y_2)$$

With boundary conditions $y_1(t_o) = \beta_1$ and $y_1\ (t_f) = \alpha_1$
or $y_2(t_o) = \beta_2$ and $y_1\ (t_f) = \alpha_2$

Table E summarizes the actions of the subroutines.

SUBROUTINE NAME	PURPOSE
RF1DI	solves non-stiff initial-value problems of any finite size using Runge-Kutta-Fehlberg algorithm.
RF3DI	Solves boundary-value problems of size two (or second order) using a modified form of Palekar algorithm .

TABLE E: A summary of subroutines for ODE

1. SUBROUTINE RF1DI

a) Purpose

The subroutine RF1DI solves a system of first order initial-value problems which are not stiff.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

```
SUBROUTINE RF1DI (N,Y,YP,T,TOUT, RE, F1, F2, F3, F4, F5, IFAIL)
```

c) Description

Before subroutine RF1DI is applied, the system of initial-value problems to be solved must be reduced to a system of first order initial-value problems. RF1DI is based on Runge-Kutta formulas developed by E. Fehlberg and implemented by L.F. Shampine and H.A. Watts. RF1DI requires six function evaluations per step. Four of these function values are combined with one set of coefficients to produce a fourth-order method, and all six values are combined with another set of coefficients to produce a fifth-order method. Comparison of the two values yields an error estimate which is used for step size control. See Forsythe et al (60) for details.

d) Parameters

N integer variable, which on entry, the user must set to the number of first order ordinary differential equations to be solved. It is not altered on exit.

Y	one-dimensional real array of size N which on entry, the user must set to the initial values of the system. On exit, it contains the solution of the system which is the values of $y_1, y_2, \cdots, y_n$ at the specified end-point TOUT.
YP	one-dimensional real array of size N, on exit YP specifies $y_1', y_2', \cdots, y_n'$ at the end point TOUT.
T	real variable which on entry specifies the initial value of the independent variable. On exit it is equal to TOUT.
TOUT	real variable which on entry specifies the final value of the independent variable. It is unaltered on exit.
RE	real variable which on entry specifies the relative error at each step of the integration. It is unaltered on exit.
F1-F5	One dimensional real <u>arrays</u>, each of size N. They are used as workspace by the subroutine.
IFAIL	integer variable, and it is set to zero on successful exit from RF1DI, otherwise it is set to a non-zero value.
<u>NOTE</u>:	All real variables must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered on exit by the subroutine can be replaced with a constant.

A subroutine which must always be named FCN must be supplied by the user and it is of the form

```
SUBROUTINE FCN (N,T,Y,YP)
DIMENSION Y(N), YP(N)
YP(1)   =  ) expression in terms of Y(1)...,Y(N) and T
YP(2)   =  )  or independent variable
  .        )
  .        )
  .        )
YP(N)   =  )
RETURN
END.
```

e) <u>Error diagnostics</u>

A successful exit from RF1DI is indicated by IFAIL equal zero

IFAIL	ERROR MESSAGE	RECOVERY
1	MAX NUMBER OF DERIVATIVE EVALUATNS EXCEEDED	Decrease the interval (T,TOUT) or increase RE. If the error persists it may be the system of equations is stiff and this subroutine is not suitable.
2	ERROR TOLERANCE TOO SMALL	Increase relative error RE.

f) <u>Other routines used</u>

RF1DI calls subroutine RF2DI. RF2DI is a part of RF1DI and it is not used by any other subroutine in the library. FUNCTION IF1MQ is referenced by RF1DI. The user is not aware of these routines.

g) Timing

The amount of time taken depends on FCN and the requested accuracy RE. The example given, took 2.0 seconds using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF1DI together with RF2DI using Tx990/4 microcomputer occupies 3,358 bytes of storage.

i) Accuracy

The result produced is accurate to the requested accuracy RE if IFAIL is zero on exit. Note that RE specifies the accuracy at each step and not the final result at TOUT. This means that, to produce a reliable final result, the system of first order initial-value problems should be well conditioned.

j) Example

RF1DI was used to solve the initial-value problem

$$y'' - 2y^3 = 0 \quad y(0) = 1,\ y'(0) = -1$$

which has exact solution $y(t) = 1.0/(1+t)$. This second order initial-value problem has to be transformed to a system of first-order initial-value problems by setting

$$y_1 = y$$

$$y_2 = y'$$

such that $y'' - 2y^3 = 0$ becomes

$$y_1' = y_2$$

$$y_2' = 2\,y_1^3$$

In this case N = 2 and FCN can now be written as shown in

example E1. The requested accuracy was 0.1 E-03 with TOUT = 3.0 and T = 0.0. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

k) Further comments

The subroutine RF1DI can still work even if T is greater than TOUT.

2. SUBROUTINE RF2DI

a) Purpose

The subroutine RF3DI solves two first-order boundary-value problems.

b) Specification

The subroutine is written in FORTRAN and it is of the form.

SUBROUTINE RF3DI (T, TOUT, TL, Y, YP, YB, ID, LMAX, IFAIL)

c) Description

Initially, the subroutine assigns the remaining initial condition to make the problem a system of two first-order initial-value problems. A modified form of RF1DI is then used to solve the system. The result obtained is compared with the boundary condition given at the end of the interval. If agreement is not attained, this result is used to update the unknown initial condition. See Palekar (128) for details.

d) Parameters

T real variable which on entry specifies the initial value of the independent variable. It is not altered on exit.

EXAMPLE E1

```
      DIMENSION Y(20),YP(20),F1(20),F2(20),F3(20),F4(20),F5(20)
      DATA IN,IOUT/20,21/
C
C  SPECIFY INITIAL CONDITIONS--
C
      T=0.0
      Y(1)=1.0
      Y(2)=-1.0
      CALL RF1DI(2,Y,YP,T,3.0,0.1E-03,F1,F2,F3,F4,F5,IFAIL)
      IF(IFAIL.NE.0) STOP
      WRITE(IOUT,100) Y(1)
100   FORMAT(4X,F6.4/)
      STOP
      END
```

```
      RESULT

    0.2499
```

```
      SUBROUTINE FCN(N,X,Y,YP)
      REAL X,Y,YP
      DIMENSION Y(N),YP(N)
      YP(1)=Y(2)
      YP(2)=2.0*Y(1)**3.0
      RETURN
      END
```

TOUT real variable which on entry specifies the final value of the independent variable. The values of the system at T and TOUT define the boundary-value problem. Not altered on exit.

TL real variable which on entry specifies the accuracy of the result to be computed.

Y real one-dimensional array of size 2. On entry Y(1) and Y(2) must be given values. One of them must be a correct value why the other an approximate value. On exit Y(1) and Y(2) contain the values of y_1 and y_2 at T respectively.

YP real one-dimensional array of size 2. On exit YP(1) contains y_2 at TOUT.

YB real variable which on entry specifies y_1 at TOUT. It is not altered on exit.

ID integer variable which on entry is set to zero if y_1 is given to be correct at T and TOUT as the boundary conditions. It should be set to 1 on entry if y_2 is given to be correct at T and y_1 is given at TOUT as the boundary conditions. It is not altered on exit.

LMAX integer variable which on entry specifies the maximum number of iterations. It is not altered on exit and a value of 3000 is suggested.

IFAIL integer variable which on successful exit from RF3DI is set to zero, otherwise it is set

to a non-zero value.

NOTE: All real variables must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered on exit by the subroutine can be replaced by a constant.

A subroutine which must always be named FCN must be supplied by the user and it is of the form:

```
SUBROUTINE FCN(N,T,Y,YP)
DIMENSION  Y(N), YP(N)
YP(1)  =  )
          ) expression in terms of Y(1), Y(2),T
YP(2)  =  )
RETURN
END.
```

e) Error diagnostics

A successful exit from RF3DI is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	MAX NUMBER OF DERIVATIVE EVALUATNS EXCEEDED	Increase TL. If it persists change the starting approximate value of Y, and try again.
2	ERROR TOLERANCE TOO SMALL	Increase TL If it persists change the starting approximate value of Y and try again.

3	DIVISION BY ZERO ENCOUNTERED	The starting approximate value must be changed.
4.	USER SUPPLIED NUMBER OF ITERATIONS EXCEEDED	Increase LMAX or TL or change starting approximate value.

f) Other routines used

RF3DI calls subroutine RF4DI and RF5DI. Both are part of RF3DI and they are not used by any other subroutine in the library. FUNCTION IF1MQ is referenced by RF3DI. The user is not aware of these routines.

g) Timing

The amount of time taken depends on FCN and the requested accuracy TL. The example given took 1.4 sec. using Tx990/4 microcomputer.

h) Storage

There are internally declared arrays of size 2. The compiled output of RF3DI together with RF4DI and RF5DI using Tx990/4 microcomputer is 4,480 bytes.

i) Accuracy

The result produced is correct to the requested accuracy TL if IFAIL is zero on exit.

j) Example

RF3DI was used to solve the boundary-value problem

$$y''+y = 0 \qquad y(0) = 0, \quad y(1) = .841471$$

which has the exact solution $y(t) = \mathrm{Sin}(t)$. This second-order initial-value problem has to be transformed to a system of first-order boundary value problems by setting

$$y_1 = y$$

$$y_2 = y_1'$$

such that $y'' + y$ becomes

$$y_1' = y_2$$

$$y_2' = -y_1$$

FCN can now be written as shown in example E2. The requested accuracy was 0.1E-03 with TOUT = 1 and T = 0.0. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

k) Further comments

If the user-supplied initial approximate value does not result in obtaining a solution, the user has to try another approximate value. On successful exit the approximate value is replaced with a value correct to the requested accuracy.

3. Listing

EXAMPLE E2

```
      DIMENSION Y(2),YP(2)
      DATA IN,IOUT/20,21/
C
C  SPECIFY INITIAL CONDITIONS-
C
      T=0.0
      TOUT=1.0
      Y(1)=0.0
      Y(2)=0.2E-02
      YB=0.841471
      ID=0
      TL=0.1E-03
      CALL RF3DI(T,TOUT,TL,Y,YP,YB,ID,3000,IFAIL)
      IF(IFAIL.NE.0) STOP
      WRITE(IOUT,200) Y(2)
200   FORMAT(4X,F6.4/)
      STOP
      END
```

RESULT

0.9999

```
      SUBROUTINE FCN(N,T,Y,YP)
      REAL T,Y,YP
      DIMENSION Y(N),YP(N)
      YP(1)=Y(2)
      YP(2)=-Y(1)
      RETURN
      END
```

SUBROUTINE RF1DI


```
      SUBROUTINE RF1DI(N,Y,YP,T,TOUT,RE,F1,F2,F3,F4,F5,IFAIL)
C.    DOUBLE PRECISION Y,YP,T,TOUT,RE,F1,F2,F3,F4,F5,DF2MQ,
C.   1   U,TL,SCALE,P1,E30ET,EE,ET,S,TL2,AE,H,HMIN,DT,YPK,ESTT
C
C  THIS SUBROUTINE SOLVES DIFFERENTIAL EAUATIONS USING RK45
C
      DIMENSION Y(N),YP(N),F1(N),F2(N),F3(N),F4(N),F5(N)
      LOGICAL HF,OUTP
      DATA P1/0.1889568E-03/
C*    DATA P1/0.1889568D-03/
      DATA MAX/3000/
      IOUT=IF1MQ(1)
      IFAIL=0
      NFE=1
      CALL FCN(N,T,Y,YP)
      IF(T.EQ.TOUT) RETURN
      TL=RF2MQ(0.0)
C*    TL=DF2MQ(0.0)
      U=0.13E 02*TL
      IF(RE.LT.TL) RE=TL
      DT=TOUT-T
C  COMPUTE INITIAL H
      H=ABS(DT)
C*    H=DABS(DT)
      DO 70 K=1,N
      TL2=RE*ABS(Y(K))+TL
C*    TL2=RE*DABS(Y(K))+TL
      YPK=ABS(YP(K))
C*    YPK=DABS(YP(K))
      IF((YPK*H**5.0).GT.TL2) H=(TL2/YPK)**0.2E 00
70    CONTINUE
      H=AMAX1(H,U*AMAX1(ABS(T),ABS(DT)))
C*    H=DMAX1(H,U*DMAX1(DABS(T),DABS(DT)))
      H=SIGN(H,DT)
C*    H=DSIGN(H,DT)
      IF(ABS(DT).GT.U*ABS(T)) GO TO 95
C  IF TOO CLOSE TO OUTPUT POINT, EXTRAPOLATE AND RETURN
C*    IF(DABS(DT).GT.(U*DABS(T))) GO TO 95
      DO 90 K=1,N
90    Y(K)=Y(K)+DT*YP(K)
      CALL FCN(N,TOUT,Y,YP)
      NFE=NFE+1
      GO TO 300
C  INITIALIZE OUTPUT POINT INDICATOR AND SCALE THE ERROR TOLORANCE
95    OUTP=.FALSE.
      SCALE=0.20E 01/RE
      AE=SCALE*TL
C  STEP BY STEP INTEGRATION
100   HF=.FALSE.
      HMIN=U*ABS(T)
```

RF1DI Cont.

PAGE 2

```
C*    HMIN=U*DABS(T)
      DT=TOUT-T
      IF(ABS(DT).GE.0.2E 01*ABS(H)) GO TO 200
C*    IF(DABS(DT).GE.(0.2D 01*ABS(H))) GO TO 200
      IF(ABS(DT).GT.ABS(H)) GO TO 150
C*    IF(DABS(DT).GT.DABS(H)) GO TO 150
      OUTP=.TRUE.
      H=DT
      GO TO 200
150   H=0.50E 00*DT
200   IF(NFE.LE.(N*MAX)) GO TO 220
      IFAIL=1
      WRITE(IOUT,500) IFAIL
500   FORMAT(I8,4X,43HMAX NUMBER OF DERIVATIVE EVALUATNS EXCEEDED/)
      RETURN
220   CALL RF2DI(N,Y,T,H,YP,F1,F2,F3,F4,F5)
C  ADVANCE AN APPROXIMATE SOLUTION OVER ONE STEP OF LENGTH H
      NFE=NFE+5
      EEDET=0.0E 00
      DO 250 K=1,N
      ET=ABS(Y(K))+ABS(F1(K))+AE
C*    ET=DABS(Y(K))+DABS(F1(K))+AE
      EE=(-0.2090E 04*YP(K)+(0.21970E 05*F3(K)-0.15046E 05*F4(K)))
     1   +(0.22526E 05*F2(K)-0.27360E 05*F5(K))
      EEDET=AMAX1(EEDET,ABS(EE)/ET)
C*    EEDET=DMAX1(EEDET,DABS(EE)/ET)
250   CONTINUE
      ESTT=ABS(H)*EEDET*SCALE/0.7524E 06
C*    ESTT=DABS(H)*EEDET*SCALE/0.7524D 06
      IF(ESTT.LE.0.1E 01) GO TO 260
      HF=.TRUE.
C  UNSUCCESSFUL STEP SIZE, REDUCE STEP SIZE AND TRY AGAIN
      OUTP=.FALSE.
      S=0.10E 00
      IF(ESTT.LT.0.59049E 05) S=0.9E 00/ESTT**0.20E 00
      H=H*S
      IF(ABS(H).GT.HMIN) GO TO 200
C*    IF(DABS(H).GT.HMIN) GO TO 200
      IFAIL=2
      WRITE(IOUT,600) IFAIL
600   FORMAT(I8,4X,25HERROR TOLORANCE TOO SMALL/)
      RETURN
C  SUCCESSFUL STEP
260   T=T+H
      DO 270 K=1,N
270   Y(K)=F1(K)
      CALL FCN(N,T,Y,YP)
      NFE=NFE+1
      S=0.5E 01
      IF(ESTT.GT.P1) S=0.9E 00/ESTT**0.2E 00
```

RF1DI AND AUX.


```
      IF(HF) S=AMIN1(S,0.1E 01)
C*    IF(HF) S=DMIN1(S,0.1D 01)
      H=SIGN(AMAX1(S*ABS(H),HMIN),H)
C*    H=DSIGN(DMAX1(S*DABS(H),HMIN),H)
      IF(.NOT.OUTP) GO TO 100
300   T=TOUT
      RETURN
      END


      SUBROUTINE RF2DI(N,Y,T,H,YP,F1,F2,F3,F4,F5)
C  FEHLBERG FOURTH-FIFTH ORDER RUNGE-KUTTA METHOD
C     DOUBLE PRECISION CH,Y,A,T,H,YP,F1,F2,F3,F4,F5,P
      DIMENSION P(12),Y(N),YP(N),F1(N),F2(N),F3(N),F4(N),F5(N)
      DATA P(1),P(2)/
     1       .93750000E-01,  .37500000E+00/
C*   1       .93750000D-01,  .37500000D+00/
      DATA P(3),P(4)/
     1       .45516614E-03,  .92307692E+00/
C*   1       .45516613563950840D-03,  .92307692307692310D+00/
      DATA P(5),P(6)/
     1       .24366472E-03,  .48732943E-04/
C*   1       .24366471734892790D-03,  .48732943469785570D-04/
      DATA P(7),P(8)/
     1       .13126719E-06,  .90288000E+06/
C*   1       .13126718779740220D-06,  .90288000D+06/
      DATA P(9),P(10)/
     1       .38557350E+07,  .13712490E+07/
C*   1       .38557350D+07,  .13712490D+07/
      DATA P(11),P(12)/
     1       .39536640E+07,  .27702000E+06/
C*   1       .39536640D+07,  .27702000D+06/
      CH=0.25E 00*H
      DO 10 K=1,N
10    F5(K)=Y(K)+CH*YP(K)
      A=T+CH
      CALL FCN(N,A,F5,F1)
      CH=H*P(1)
      DO 20 K=1,N
20    F5(K)=Y(K)+CH*(YP(K)+0.30E 01*F1(K))
      A=T+P(2)
      CALL FCN(N,A,F5,F2)
      CH=H*P(3)
      DO 30 K=1,N
30    F5(K)=Y(K)+CH*(0.1932E 04*YP(K)+(0.7296E 04*F2(K)-0.72E 04*F1(K)))
      A=T+H*P(4)
      CALL FCN(N,A,F5,F3)
      CH=H*P(5)
      DO 40 K=1,N
      F5(K)=Y(K)+CH*((0.8341E 04*YP(K)-0.8454E 03*F3(K))+
```

RF1DI AND AUX. Cont.


```
     1            (0.29440E 05*F2(K)-0.32832E 05*F1(K)))
 40   CONTINUE
      A=T+H
      CALL FCN(N,A,F5,F4)
      CH=H*P(6)
      DO 50 K=1,N
      F1(K)=Y(K)+CH*((-0.6080E 04*YP(K)+(0.9295E 04*F3(K)-
     1      0.5643E 04*F4(K)))+(0.41040E 05*F1(K)-0.28352E 05*F2(K)))
 50   CONTINUE
      A=T+0.50E 00*H
      CALL FCN(N,A,F1,F5)
C  COMPUTE APPROXIMATE SOLUTION AT T+H
      CH=H*P(7)
      DO 60 K=1,N
      F1(K)=Y(K)+CH*((P(8)*YP(K)+(P(9)*F3(K)-P(10)*F4(K)))+
     1      (P(11)*F2(K)+P(12)*F5(K)))
 60   CONTINUE
      RETURN
      END
```

SUBROUTINE RF3DI


```
      SUBROUTINE RF3DI(T,TOUT,TL,Y,YP,YB,ID,LMAX,IFAIL)
C  THIS SUBROUTINE SOLVES A SYSTEM OF TWO FIRST ORDER
C  TWO-POINT BOUNDARY VALUE PROBLEMS USING PALEKAR'S ALGORITHM
C.    DOUBLE PRECISION TL,TL2,Y,YB,YP,YT,C11,
C.   1    C21,P11,P21,D11,D12,A,T,TOUT,XT,DET,TL3
      DIMENSION Y(2),YP(2),YT(2)
C  INITIALIZE VALUES
      TL2=0.5E 00*TL
      TL3=YB*TL+TL2
      IOUT=IF1MQ(1)
      IF(ID.EQ.0) GO TO 10
      C11=Y(2)
      C21=Y(1)
      GO TO 20
10    C11=Y(1)
      C21=Y(2)
20    YT(1)=Y(1)
      YT(2)=Y(2)
C  INTEGRATE WITH APPROXIMATE INITIAL CONDITIONS
      CALL RF4DI(YT,YP,A,TOUT,TL2,IFAIL)
      IF(IFAIL.NE.0) RETURN
      P11=YT(1)
C  CHECK FOR CONVERGENCE
      IF(ABS(YB-P11).LT.TL3) RETURN
C*    IF(DABS(YB-P11).LT.TL3) RETURN
      IF(ID.EQ.0) GO TO 30
      YT(1)=C21
      YT(2)=Y(2)
      Y(1)=C11
      GO TO 40
30    Y(2)=C11
      YT(1)=Y(1)
      YT(2)=C21
40    A=T
      CALL RF4DI(YT,YP,A,TOUT,TL2,IFAIL)
      IF(IFAIL.NE.0) RETURN
      P21=YT(1)
      IF(ID.EQ.0) Y(2)=C21
      IF(ID.NE.0) Y(1)=C21
C  CHECK FOR CONVERGENCE
      IF(ABS(YB-P21).LT.TL3) RETURN
C*    IF(DABS(YB-P21).LT.TL3) RETURN
      L=0
45    L=L+1
      DET=C11-C21
      IF(DET.EQ.0.0E 00) GO TO 90
      DET=0.1E 01/DET
      D11=(P11-P21)*DET
      D12=(P21*C11-P11*C21)*DET
      IF(D11.EQ.0.0E 00) GO TO 90
      XT=(YB-D12)/D11
```

RF3DI AND AUX.


```
      IF(ID.EQ.0) GO TO 50
      YT(1)=XT
      YT(2)=Y(2)
      GO TO 60
50    YT(1)=Y(1)
      YT(2)=XT
60    A=T
      CALL RF4DI(YT,YP,A,TOUT,TL2,IFAIL)
      IF(IFAIL.NE.0) RETURN
      IF(ID.EQ.0) Y(2)=XT
      IF(ID.NE.0) Y(1)=XT
      IF(ABS(YB-YT(1)).LT.TL3) RETURN
C*    IF(DABS(YB-YT(1)).LT.TL3) RETURN
      IF(L.EQ.(2*(L/2))) GO TO 70
      C11=XT
      P11=YT(1)
      GO TO 80
70    C21=XT
      P21=YT(1)
C  TEST WHETHER MAXIMUM NUMBER OF ITERATIONS HAVE BEEN PERFORMED
80    IF(L.LT.LMAX) GO TO 45
      IFAIL=4
      WRITE(IOUT,100) IFAIL
100   FORMAT(I8,4X,43HUSER SUPPLIED NUMBER OF ITERATIONS EXCEEDED/)
      RETURN
90    IFAIL=3
      WRITE(IOUT,200) IFAIL
200   FORMAT(I8,4X,28HDIVISION BY ZERO ENCOUNTERED/)
      RETURN
      END


      SUBROUTINE RF4DI(Y,YP,T,TOUT,RE,IFAIL)
C.    DOUBLE PRECISION Y,YP,T,TOUT,RE,F1,F2,F3,F4,F5,DTOL,
C.   1  U,TL,SCALE,P1,EEDET,EE,ET,S,TL2,AE,H,HMIN,DT,YPK,ESTT
C
C  THIS SUBROUTINE SOLVES DIFFERENTIAL EQUATIONS USING RK45
C
      DIMENSION Y(2),YP(2),F1(2),F2(2),F3(2),F4(2),F5(2)
      LOGICAL HF,OUTP
      DATA P1/0.1889568E-03/
C*    DATA P1/0.1889568D-03/
      DATA MAX/3000/
      IOUT=IF1MQ(1)
      IFAIL=0
      NFE=1
      N=2
      CALL FCN(N,T,Y,YP)
      IF(T.EQ.TOUT) RETURN
      TL=RF2MQ(0.0)
C*    TL=DF2MQ(0.0)
```

RF3DI AND AUX. Cont.


```
      U=0.13E 02*TL
      IF(RE.LT.TL) RE=TL
      DT=TOUT-T
C  COMPUTE INITIAL H
      H=ABS(DT)
C*    H=DABS(DT)
      DO 70 K=1,N
      TL2=RE*ABS(Y(K))+TL
C*    TL2=RE*DABS(Y(K))+TL
      YPK=ABS(YP(K))
C*    YPK=DABS(YP(K))
      IF((YPK*H**5.0).GT.TL2) H=(TL2/YPK)**0.2E 00
70    CONTINUE
      H=AMAX1(H,U*AMAX1(ABS(T),ABS(DT)))
C*    H=DMAX1(H,U*DMAX1(DABS(T),DABS(DT)))
      H=SIGN(H,DT)
C*    H=DSIGN(H,DT)
      IF(ABS(DT).GT.U*ABS(T)) GO TO 95
C  IF TOO CLOSE TO OUTPUT POINT, EXTRAPOLATE AND RETURN
C*    IF(DABS(DT).GT.(U*DABS(T))) GO TO 95
      DO 90 K=1,N
90    Y(K)=Y(K)+DT*YP(K)
      CALL FCN(N,TOUT,Y,YP)
      NFE=NFE+1
      GO TO 300
C  INITIALIZE OUTPUT POINT INDICATOR AND SCALE THE ERROR TOLORANCE
95    OUTP=.FALSE.
      SCALE=0.20E 01/RE
      AE=SCALE*TL
C  STEP BY STEP INTEGRATION
100   HF=.FALSE.
      HMIN=U*ABS(T)
C*    HMIN=U*DABS(T)
      DT=TOUT-T
      IF(ABS(DT).GE.0.2E 01*ABS(H)) GO TO 200
C*    IF(DABS(DT).GE.(0.2D 01*ABS(H))) GO TO 200
      IF(ABS(DT).GT.ABS(H)) GO TO 150
C*    IF(DABS(DT).GT.DABS(H)) GO TO 150
      OUTP=.TRUE.
      H=DT
      GO TO 200
150   H=0.50E 00*DT
200   IF(NFE.LE.(N*MAX)) GO TO 220
      IFAIL=1
      WRITE(IOUT,500) IFAIL
500   FORMAT(I8,4X,43HMAX NUMBER OF DERIVATIVE EVALUATNS EXCEEDED/)
      RETURN
220   CALL RF5DI(N,Y,T,H,YP,F1,F2,F3,F4,F5)
C  ADVANCE AN APPROXIMATE SOLUTION OVER ONE STEP OF LENGTH H
      NFE=NFE+5
      EEDET=0.0E 00
```

RF3DI AND AUX. Cont.


```
      DO 250 K=1,N
      ET=ABS(Y(K))+ABS(F1(K))+AE
C*    ET=DABS(Y(K))+DABS(F1(K))+AE
      EE=(-0.2090E 04*YP(K)+(0.21970E 05*F3(K)-0.15046E 05*F4(K)))
     1   +(0.22528E 05*F2(K)-0.27340E 05*F5(K))
      EEOET=AMAX1(EEOET,ABS(EE)/ET)
C*    EEOET=DMAX1(EEOET,DABS(EE)/ET)
250   CONTINUE
      ESTT=ABS(H)*EEOET*SCALE/0.7524E 06
C*    ESTT=DABS(H)*EEOET*SCALE/0.7524D 06
      IF(ESTT.LE.0.1E 01) GO TO 260
      HF=.TRUE.
C  UNSUCCESSFUL STEP SIZE, REDUCE STEP SIZE AND TRY AGAIN
      OUTP=.FALSE.
      S=0.10E 00
      IF(ESTT.LT.0.59049E 05) S=0.9E 00/ESTT**0.20E 00
      H=H*S
      IF(ABS(H).GT.HMIN) GO TO 200
C*    IF(DABS(H).GT.HMIN) GO TO 200
      IFAIL=2
      WRITE(IOUT,600) IFAIL
600   FORMAT(I8,4X,25HERROR TOLORANCE TOO SMALL/)
      RETURN
C  SUCCESSFUL STEP
260   T=T+H
      DO 270 K=1,N
270   Y(K)=F1(K)
      CALL FCN(N,T,Y,YP)
      NFE=NFE+1
      S=0.5E 01
      IF(ESTT.GT.P1) S=0.9E 00/ESTT**0.2E 00
      IF(HF) S=AMIN1(S,0.1E 01)
C*    IF(HF) S=DMIN1(S,0.1D 01)
      H=SIGN(AMAX1(S*ABS(H),HMIN),H)
C*    H=DSIGN(DMAX1(S*DABS(H),HMIN),H)
      IF(.NOT.OUTP) GO TO 100
300   T=TOUT
      RETURN
      END


      SUBROUTINE RF5DI(N,Y,T,H,YP,F1,F2,F3,F4,F5)
C  FEHLBERG FOURTH-FIFTH ORDER RUNGE-KUTTA METHOD
C.    DOUBLE PRECISION CH,Y,A,T,H,YP,F1,F2,F3,F4,F5,P
      DIMENSION P(12),Y(N),YP(N),F1(N),F2(N),F3(N),F4(N),F5(N)
      DATA P(1),P(2)/
     1       .93750000E-01,  .37500000E+00/
C*   1       .93750000D-01,  .37500000D+00/
      DATA P(3),P(4)/
     1       .45516614E-03,  .92307692E+00/
C*   1       .455166135639508 4D-03,  .9230769230769231D+00/
```

RF3DI AND AUX. Cont.


```
      DATA P(5),P(6)/
     1       .24366472E-03,  .48732943E-04/
C*   1       .2436647173489279D-03,  .4873294346978557D-04/
      DATA P(7),P(8)/
     1       .13126719E-06,  .90288000E+06/
C*   1       .1312671877974022D-06,  .90288000D+06/
      DATA P(9),P(10)/
     1       .38557350E+07,  .13712490E+07/
C*   1       .38557350D+07,  .13712490D+07/
      DATA P(11),P(12)/
     1       .39536640E+07,  .27702000E+06/
C*   1       .39536640D+07,  .27702000D+06/
      CH=0.25E 00*H
      DO 10 K=1,N
10    F5(K)=Y(K)+CH*YP(K)
      A=T+CH
      CALL FCN(N,A,F5,F1)
      CH=H*P(1)
      DO 20 K=1,N
20    F5(K)=Y(K)+CH*(YP(K)+0.30E 01*F1(K))
      A=T+P(2)
      CALL FCN(N,A,F5,F2)
      CH=H*P(3)
      DO 30 K=1,N
30    F5(K)=Y(K)+CH*(0.1932E 04*YP(K)+(0.7296E 04*F2(K)-0.72E 0 *F1(K)))
      A=T+H*P(4)
      CALL FCN(N,A,F5,F3)
      CH=H*P(5)
      DO 40 K=1,N
      F5(K)=Y(K)+CH*((0.8341E 04*YP(K)-0.845E 03*F3(K))+
     1              (0.29440E 05*F2(K)-0.32832E 05*F1(K)))
40    CONTINUE
      A=T+H
      CALL FCN(N,A,F5,F4)
      CH=H*P(6)
      DO 50 K=1,N
      F1(K)=Y(K)+CH*((-0.6080E 04*YP(K)+(0.9295E 04*F3(K)-
     1      0.5643E 04*F4(K)))+(0.41040E 05*F1(K)-0.26352E 05*F2(K)))
50    CONTINUE
      A=T+0.50E 00*H
      CALL FCN(N,A,F1,F5)
C  COMPUTE APPROXIMATE SOLUTION AT T+H
      CH=H*P(7)
      DO 60 K=1,N
      F1(K)=Y(K)+CH*((P(8)*YP(K)+(P(9)*F3(K)-P(10)*F4(K)))+
     1      (P(11)*F2(K)+P(12)*F5(K)))
60    CONTINUE
      RETURN
      END
```

APPENDIX F

MINIMIZATION AND LEAST SQUARES APPROXIMATION

The subroutines documented in this appendix are used to:

i) minimize nonlinear functions of single or several variables.

ii) approximate a given set of data (co-ordinates) with a polynomial using the method of least squares.

Table F summarizes the action of the subroutines.

SUBROUTINE NAME	PURPOSE
RF1MI	Determines the local minimum in an interval (a,b) of a non-linear single variable function using Brent's algorithm.
RF2MI	Determines the local minimum of a nonlinear function of several variables using Rosenbrock algorithm.
RF1LS	Fits least squares polynomial to a set of given data points.

TABLE F: A summary of subroutines for minimization and approximation.

1. SUBROUTINE RF1MI

a) Purpose

Subroutine RF1MI locates the minimum of a nonlinear function of single variable in a given interval.

b) Specification

The subroutine is written in FORTRAN and it is of the form:

SUBROUTINE RF1MI(AX, BX, FUNCT, FMIN, EPS)

c) Description

The method used is based on safe gaurded successive-interpolation schemes. It combines the guaranteed convergence attribute of the golden-section search with the superlinear asymptotic rate of convergence of successive parabolic interpolation. (see Forsythe et al (60) and Brent (15)).

d) Parameters

AX — real variable which on entry the user must set to the lower limit of the interval containing the minimum to be determined. It is unaltered on exit.

BX — real variable which on entry, the user must set to the upper limit of the interval containing the minimum to be determined. It is not altered on exit.

FUNCT — real function supplied by the user. It defines the function whose minimum is required in the given interval (AX, BX).

It's specification is:

REAL FUNCTION FUNCT(X)

REAL X

FUNCT must be declared as EXTERNAL in the calling (sub) program.

FMIN real variable, on exit, FMIN approximates the point where FUNCT attains a minimum in the interval (AX, BX).

EPS real variable, on entry, it specifies the relative accuracy to which the minimum is required.

e) Error diagnostics

None

f) Other routines used

None

g) Timing

The amount of time taken depends on the function whose minimum is required and the size of the interval containing a minimum. For the function given in the example, it took 0.35 sec to compute a minimum using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF1MI using Tx990/4 microcomputer is 1,568 bytes.

i) Accuracy

Result is accurate to the requested accuracy or machine precision depending on which one is larger.

j) Example

RF1MI was used to locate a minimum of the function:

$$f(x) = x^3-2x-5$$

in the interval (0,1),

and the requested accuracy was 0.94E-05. Example F1 shows how RF1MI can be used. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

2. SUBROUTINE RF2MI

a) Purpose

Subroutine RF2MI locates the local minimum of a nonlinear function of several variables which is not constrained.

b) Specification

The subroutine is written in FORTRAN and it is of the form

```
      SUBROUTINE RF2MI (X, FUNCT, EPS, F, ID, N, V,
     1                  ALPHA, A,D,E, IFAIL)
```

c) Description

The local minimum of a function is sought by conducting cyclic searches parallel to each of the orthogonal unit vectors, the coordinate directions, in turn. n (where n is the number of variables) such searches constitute one stage of the iteration process. For the next stage a set of n orthogonal unit vectors is generated, such that the first vector of this set lies along the direction of greatest advance for the previous stage. The Gram - Schmidt orthogonalization procedure is used to calculate the new unit vectors. (see Rosenbrock (142) and Machura et al (107)).

EXAMPLE F1

```
      REAL AX,BX,EPS,FMIN
      EXTERNAL FUN
      DATA IN,IOUT/20,21/
C  SPECIFY INTERVAL ENCLOSING A MINIMUM
      AX=0.0
      BX=1.0
      EPS=0.9405-05
      CALL RF1MI(AX,BX,FUN,FMIN,EPS)
      WRITE(IOUT,200) FMIN
200   FORMAT(E14.7)
      STOP
      END
```

RESULT

```
0.6165544E 00
```

```
      REAL FUNCTION FUN(X)
      FUN=X*X*X-2.0*X-5.0
      RETURN
      END
```

d) Parameters

X — real one-dimensional array of size N where N is the number of variables. On entry X is the initial estimate of the solution. On exit X is the best estimate of the solution found to the requested accuracy.

FUNCT — real function, supplied by the user. It defines the function whose local minimum is to be computed. It's specification is

```
REAL FUNCTION F(N,X)
REAL X(N)
```

FUNCT must be declared as EXTERNAL in the (sub) program from which RF2MI is called.

EPS — real variable, on entry, it specifies the relative accuracy to which the solution is required. It is unaltered on exit. If a user supplied EPS is less than 5 * machine relative error, then EPS is assigned a value of Square root (5 * machine relative error). It is suggested that EPS should be chosen to be equal to SQRT(machine relative error).

F — real variable which on exit gives the value of the Function FUNCT at the solution X.

ID — integer variable, which on entry, the user must set to the row size of matrix V and ALPHA declared in the calling program. This means that V and ALPHA must always have the same row size in the calling (sub) program. It is not altered on exit.

N	integer variable which on entry, the user must set to the number of variables of the functions to be minimized. It is unaltered on exit.
V	real two-dimensional array of size NxM which is used as workspace, with $M \geq N$.
ALPHA	real two-dimensional array of size N x M which is used as workspace with $M \geq N$
A	real one-dimensional array of size N used as workspace.
D	real one-dimensional array of size N used as workspace.
E	real one-dimensional array of size N used as workspace.
IFAIL	integer variable, it is set to zero on successful exit from RF2MI, otherwise it is set to a non-zero value.
NOTE:	All real variables must be declared DOUBLE PRECISION if the double precision version of the library is implemented. Also any variable which is not altered by the subroutine can be replaced with a constant.

e) Error diagnostics

A successful return from RF2MI is indicated by IFAIL equal zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	CONV NOT ACHIEVED AFTER 400*N FUN EVALUATIONS	The user can call the subroutine again using the present solution as initial estimate or EPS can be increased and the subroutine called again.

2	NUMBER OF VARIABLES LESS THAN 2	If N = 1, then subroutine RF1MI should be used.
3	ITERATION STUCK ON ONE SIDE	This usually happens when EPS is too small or the initial estimate is too far from the solution. Use a new initial estimate or increase EPS and use the solution obtained in the first call as your initial estimate. In most cases it is the best solution RF2MI can obtain, when IFAIL = 3 on exit. It is not likely to improve on this solution on subsequent calls.

f) Other routines used

The function IF1MQ is referenced.

g) Timing

Depends on the function, the initial point and the requested accuracy. For the example given, RF2MI took 5.5 sec. using Tx990/4 microcomputer.

h) Storage

There are no internally declared arrays and the compiled output of RF2MI using Tx990/4 microcomputer is 2,598 bytes.

i) Accuracy

Depends on EPS.

j) Example

Example F2 shows how the subroutine RF2MI can be used. In this

EXAMPLE F2

```
      DIMENSION V(10,10),ALP(10,10),A(10),D(10),E(10),X(10)
      DATA IN,IOUT/20,21/
C  INPUT NUMBER OF VARIABLES
      READ(IN,300) N
C  INPUT INITIAL VALUE OF X
      READ(IN,100)(X(I),I=1,N)
      CALL RF2MI(X,ROSEN,EPS,F,10,N,V,ALP,A,D,E,IFAIL)
      IF(IFAIL.NE.0) STOP
      WRITE(IOUT,100)(X(I),I=1,N)
      WRITE(IOUT,200) F
100   FORMAT(2E14.7)
200   FORMAT(E14.7)
300   FORMAT(I4)
      STOP
      END
```

DATA

```
  2
 -.1200000E 01 0.1000000E 01
```

RESULT

```
 0.9999368E 00 0.9998747E 00
 0.3969504E-03
```

FUNCTION WHOSE MINIMUM IS DETERMINED

```
      REAL FUNCTION ROSEN(N,X)
      DIMENSION X(N)
C
C  THIS IS ROSEN
C
50    ROSEN=0.100E 03*(X(2)-X(1)*X(1))*(X(2)-X(1)*X(1))
     1  +(0.10E 01-X(1))*(0.10E 01-X(1))
      RETURN
      END
```

example the local minimum of the function:

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

is located with initial point taken as (-1.2, 1). This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

3. SUBROUTINE RF1LS

a) Purpose

RF1LS fits a least squares polynomial to a given set of co-ordinates.

b) Specification

RF1LS is written in FORTRAN and it is of the form:

```
SUBROUTINE RF1LS (F,X,W,M,K1,SI,F,RLP,TP,X2,L)
```

c) Description

This method is based on the use of orthogonal polynomials. (see Makinson (109) and Forsythe (58) for more details)

d) Parameters

F real one-dimensional array of size M. On entry, the array F specifies the ordinates. Not altered on exit.

X real one-dimensional array of size M. On entry, the array X specifies the abscissas; X(1) should be algebraically the smallest abscissa and X(M) the largest. Not altered on exit.

W real one-dimensional array of size M. On entry the array W specifies the weights which should be provided inversely proportional to the square of the standard error of the observations.

If no weights are to be used, then W(k) = 1.0 , K = 1, ····, M before entry to subroutine. It is unaltered on exit.

M integer variable which on entry the user must set to the number of coordinates. Not altered on exit.

K1 integer variable which on entry the user must set to the number of coefficients of the interpolating polynomial (degree of polynomial + 1) to be computed. K1 < M

P real one-dimensional array of size K1. On exit P contains the coefficients of the computed interpolating polynomial, with P(1) the independent term.

SI real one-dimensional array of size K1. On exit, SI contains the measures of the goodness of fit of each polynomial tested starting from degree 1 to K1-1.

RLP real one-dimensional array of size M, used as work space.

TF real one-dimensional array of size M used as workspace.

X2 real one-dimensional array of size M used as workspace.

L logical variable, which before entry, the user must set to.FALSE. or.TRUE. If L is .FALSE. it means that a polynomial with less number of

coefficients than K1 can be chosen if it is found to be equally good a fit as a polynomial with K1 coefficients. If L is .TRUE. it means the polynomial must have exactly K1 coefficients. It is not altered on exit.

e) Error diagnostics

None

f) Other routines used

The subroutine RF2LS is used. It is a part of RF1LS. RF2LS is not used by any other routine in the library and it is not available to the user.

g) Timing

Depends on the number of co-ordinates and the degree of the polynomial to be obtained. For the example given, Tx990/4 microcomputer used 0.92 sec.

h) Storage

There are no internally declared arrays and the compiled output of RF1LS together with RF2LS using Tx990/4 microcomputer occupies 2,570 bytes.

i) Accuracy

Depends on the behaviour of the function to be interpolated. The subroutine does not give any estimate of the error in the interpolating polynomial obtained.

j) Example

Example F3 shows how RF1LS can be used in this example RF1LS is used to fit a polynomial of degree 5 to a set of 10 data points. The polynomial obtained is evaluated at the point

0.55. The points given represent the function $f(x) = \sqrt{x}$. This single precision example requires amendment for use in DOUBLE PRECISION implementation and the result produced may differ slightly.

4. LISTINGS

Example F3

```
      DIMENSION F(20),X(20),X2(20),W(20),RLP(20),TP(20),SI(11),
     *          P(11)
      DIMENSION K(3)
      LOGICAL L
      DATA IN,IOUT/20,21/
C  INITIALIZE VALUES
      L=.FALSE.
      READ(20,100) M
      DO 10 K=1,M
      READ(20,200) X(K),F(K),W(K)
10    CONTINUE
C  CALL SUBROUTINE
      CALL RF1LB(F,X,W,M,K1,SI,P,RLP,TP,X2,L)
C  EVALUATE POLYNOMIAL AT A GIVEN POINT
      POINT=0.55E 00
      K2=K1-1
      SPK=P(K1)
      DO 40 I=1,K2
      J=K1-I
40    SPK=SPK*POINT+P(J)
      WRITE(IOUT,400) SPK,K1
      WRITE(IOUT,600)(P(J),J=1,K1)
600   FORMAT(3E14.7)
300   FORMAT(E14.7)
100   FORMAT(I4)
200   FORMAT(3E14.7)
400   FORMAT(E14.7,I4)
      STOP
      END
```

DATA

```
  10
  .1000000E+00  .3162278E+00  .1000000E+01
  .2000000E+00  .4472136E+00  .1000000E+01
  .3000000E+00  .5477226E+00  .1000000E+01
  .4000000E+00  .6324555E+00  .1000000E+01
  .5000000E+00  .7071068E+00  .1000000E+01
  .6000000E+00  .7745967E+00  .1000000E+01
  .7000000E+00  .8366600E+00  .1000000E+01
  .8000000E+00  .8944272E+00  .1000000E+01
  .9000000E+00  .9486833E-00  .1000000E+01
  .1000000E+01  .1000000E+01  .1000000E+01
```

RESULTS

```
  .7412733E 00   6
  .1394601E+00  .2077346E+01 -.3527716E+01  .4770710E+01 -.3485556E+01
  .1025849E+01
```

SUBROUTINE RF1MI

PAGE 1

```
      SUBROUTINE RF1MI(AX,BX,FUNCT,FMIN,EPS)
C.    DOUBLE PRECISION A,B,FUNCT,FMIN,EPS,AX,BX,C,D,E,DF2MQ,TL,
C.   1                 XM,P,Q,R,TOL1,TOL2,U,V,W,FU,FV,FW,FX,X
C  C IS THE SQUARED INVERSE OF THE GOLDEN RATIO
      C=0.500E 00*(0.30E 01-SQRT(0.50E 01))
C*    C=0.50D 00*(0.30D 01-DSQRT(0.50D 01))
      TL=RF2MQ(0.0)
C*    TL=DF2MQ(0.0)
      TL=SQRT(TL)
C*    TL=DSQRT(TL)
C  INITIALIZATION
      A=AX
      B=BX
      V=A+C*(B-A)
      W=V
      X=V
      E=0.0
      FX=FUNCT(X)
      FV=FX
      FW=FX
C  MAIN LOOP STARTS HERE
20    XM=0.50E 00*(A+B)
      TOL1=TL*ABS(X)+0.3333333E 00*EPS
C*    TOL1=TL*DABS(X)+0.3333333333333333D 00*EPS
      TOL2=0.20E 01*TOL1
      IF(ABS(X-XM).LE.(TOL2-0.50E 00*(B-A))) GO TO 90
C*    IF(DABS(X-XM).LE.(TOL2-0.50D 00*(B-A))) GO TO 90
      IF(ABS(E).LE.TOL1) GO TO 40
C*    IF(DABS(E).LE.TOL1) GO TO 40
      R=(X-W)*(FX-FV)
      Q=(X-V)*(FX-FW)
      P=(X-V)*Q-(X-W)*R
      Q=0.20E 01*(Q-R)
      IF(Q.GT.0.0) P=-P
      Q=ABS(Q)
C*    Q=DABS(Q)
      R=E
      E=D
30    IF(ABS(P).GE.ABS(0.50E 00*Q*R)) GO TO 40
C* 30 IF(DABS(P).GE.DABS(0.50D 00*Q*R)) GO TO 40
      IF(P.LE.Q*(A-X)) GO TO 40
      IF(P.LE.A*(B-X)) GO TO 40
C  A PARABOLIC INTERPOLATION STEP
      D=P/Q
      U=X+D
      IF((U-A).LT.TOL2) D=SIGN(TOL1,XM-X)
C*    IF((U-A).LT.TOL2) D=DSIGN(TOL1,XM-X)
      IF((B-U).LT.TOL2) D=SIGN(TOL1,XM-X)
C*    IF((B-U).LT.TOL2) D=DSIGN(TOL1,XM-X)
      GO TO 50
```

RF1MI Cont.


```
40    IF(X.GE.XM) E=A-X
      IF(X.LT.XM) E=B-X
      D=C*E
50    IF(ABS(D).GE.TOL1) U=X+D
C* 50 IF(DABS(D).GE.TOL1) U=X+D
      IF(ABS(D).LT.TOL1) U=X+SIGN(TOL1,D)
C*    IF(DABS(D).LT.TOL1) U=X+DSIGN(TOL1,D)
      FU=FUNCT(U)
C  UPDATE A,B,V,W, AND X
      IF(FU.GT.FX) GO TO 60
      IF(U.GE.X) A=X
      IF(U.LT.X) B=X
      V=W
      FV=FW
      X=U
      FX=FU
      GO TO 20
60    IF(U.LT.X) A=U
      IF(U.GE.X) B=U
      IF(FU.LE.FW) GO TO 70
      IF(W.EQ.X) GO TO 70
      IF(FU.LE.FV) GO TO 80
      IF(V.EQ.X) GO TO 80
      IF(V.EQ.W) GO TO 80
      GO TO 20
70    V=W
      FV=FW
      W=U
      FW=FU
      GO TO 20
80    V=U
      FV=FU
      GO TO 20
C  END OF MAIN LOOP
90    FMIN=X
      RETURN
      END
```

SUBROUTINE RF2MI


```
      SUBROUTINE RF2MI(X,FUNCT,EPS,F,ID,N,V,ALPHA,A,D,E,IFAIL)
C     DOUBLE PRECISION X,EPS,EPS2,FUNCT,A,D,V,ALPHA,BETA,E,F0,
C    2                 F1,B,DF2MQ,BETY
C-    DOUBLE PRECISION Q
      DIMENSION V(ID,N),ALPHA(ID,N),A(N),D(N),E(N),X(N)
      IOUT=IF1MQ(1)
      IF(N.GT.1) GO TO 5
      IFAIL=2
      WRITE(IOUT,500) IFAIL
500   FORMAT(I8,4X,31HNUMBER OF VARIABLES LESS THAN 2/)
      RETURN
5     NCY=1
      IF(N.GT.4) NCY=2
      IF(N.GT.10) NCY=3
      IFAIL=0
      TL=0.50E 01*RF2MQ(0.0)
C*    TL=0.100E 02*DF2MQ(0.0)
      IF(EPS.LT.TL) EPS=SQRT(TL)
C#    IF(EPS.LT.TL) EPS=DSQRT(TL)
      EPS2=FLOAT(N)*EPS
C*    EPS2=DBLE(FLOAT(N))*EPS
      DO 10 I=1,N
10    E(I)=0.1000E 00
      NFV=0
C  V(I,J) IS A  N*N MATRIX DEFINING A SET OF N MUTUALLY
C  ORTHOGONAL COORDINATE DIRECTIONS. V(I,J) IS THE UNIT
C  MATRIX AT THE START OF THE PROCESS
      DO 30 I=1,N
      DO 20 J=1,N
20    V(I,J)=0.0
30    V(I,I)=0.100E 01
      MTK=400*N
      F0=FUNCT(N,X)
C  START OF ITERATION LOOP
40    DO 50 I=1,N
      A(I)=0.200E 01
50    D(I)=0.0
C  EVALUATE F AT THE NEW POINT X
      K=0
60    K=K+1
      IF(K.LT.30) GO TO 65
C  ITERATION STUCK ON ONE SIDE
      IFAIL=3
      DO 68 L=1,N
68    X(L)=ALPHA(1,L)
      F=B
      WRITE(IOUT,600) IFAIL
600   FORMAT(I8,4X,27HITERATION STUCK ON ONE SIDE/)
      RETURN
65    DO 130 I=1,N
```

RF2MI Cont.


```
      DO 70 J=1,N
70    X(J)=X(J)+E(I)*V(I,J)
      NFV=NFV+1
      IF(NFV.GT.MTK) GO TO 290
      F1=FUNCT(N,X)
      IF(F1-F0) 80,90,90
C  NEW VALUE OF THE FUNCTION IS LESS THAN THE OLD ONE
80    D(I)=D(I)+E(I)
      E(I)=0.30E 01*E(I)
      F0=F1
      IF(A(I).GT.0.15E 01) A(I)=0.10E 01
      GO TO 110
C  THE NEW VALUE OF THE FUNCTION IS GREATER THAN OR EQUAL
C  THE OLD ONE
90    DO 100 J=1,N
100   X(J)=X(J)-E(I)*V(I,J)
      E(I)=-0.500E 00*E(I)
      IF(A(I).LT.0.15E 01) A(I)=0.0
110   DO 120 J=1,N
      IF(A(J).GE.0.500E 00) GO TO 130
120   CONTINUE
      GO TO 140
130   CONTINUE
      GO TO 60
C  GRAM-SCHMIDT ORTHOGONILISATION PROCESS
140   DO 160 I=1,N
      DO 160 J=1,N
      Q=0.0
      DO 150 L=I,N
150   Q=Q+D(L)*V(L,J)
C+150 Q=Q+DBLE(D(L))*V(L,J)
160   ALPHA(I,J)=Q
C+160 ALPHA(I,J)=SNGL(Q)
      Q=0.0
      DO 200 J=1,N
200   Q=Q+ALPHA(1,J)*ALPHA(1,J)
C+200 Q=Q+DBLE(ALPHA(1,J))*ALPHA(1,J)
      B=Q
C+    B=SNGL(Q)
      B=SQRT(B)
C*    B=DSQRT(B)
      L=0
C  TEST FOR CONVERGENCE
      DO 205 J=1,N
      IF(B.GT.(EPS2*ABS(X(J))+TL)) L=1
C*    IF(B.GT.(EPS2*DABS(X(J))+TL)) L=1
205   CONTINUE
      IF(L.EQ.0) GO TO 300
C  CALCULATE THE NEW SET OF ORTHONORMAL COORDINATE
C  DIRECTIONS ( THE NEW MATRIX V(I,J) )
```

RF2MI Cont.

PAGE 3

```
      DO 210 J=1,N
210   V(1,J)=ALPHA(1,J)/B
      DO 280 L=1,NCY
      DO 250 MP=2,N
      IP=MP-1
      DO 230 M=MP,N
      Q=0.0E 00
      DO 220 K=1,N
220   Q=Q-ALPHA(M,K)*V(IP,K)
C+220 Q=Q-DBLE(ALPHA(M,K))*V(IP,K)
      BETY=Q
C+    BETY=SNGL(Q)
      DO 230 J=1,N
230   ALPHA(M,J)=ALPHA(M,J)+BETY*V(IP,J)
      Q=0.0E 00
      DO 240 K=1,N
240   Q=Q+ALPHA(MP,K)*ALPHA(MP,K)
C+240 Q=Q+DBLE(ALPHA(MP,K))*ALPHA(MP,K)
      BETY=Q
C+    BETY=SNGL(Q)
      BETY=SQRT(BETY)
C*    BETY=DSQRT(BETY)
      DO 250 K=1,N
250   V(MP,K)=ALPHA(MP,K)/BETY
      IF(L.EQ.NCY) GO TO 280
      DO 260 I=2,N
      DO 260 J=1,N
260   ALPHA(I,J)=V(I,J)
280   CONTINUE
C  END OF GRAM-SCHMIDT PROCESS
      DO 285 L=1,N
285   ALPHA(1,L)=X(L)
      B=F0
      GO TO 40
C  GO TO NEXT ITERATION
290   IFAIL=1
      WRITE(IOUT,700) IFAIL
700   FORMAT(I8,4X,43HCONV NOT ACHIEVED AFTER 400*N FUN EVALUATNS/)
300   F=F0
      RETURN
      END
```

SUBROUTINE RF1LS


```
      SUBROUTINE RF1LS(F,X,W,M,K1,SI,P,RLP,TP,X2,L)
C.    DOUBLE PRECISION F,W,X,X2,RLP,TP,SI,S,P,AL,B0,CTP,CPS,CP,
C.   1                 CLP,SIMIN,DELSQ,A,B,TW,OM,DU,RLW,B1
      LOGICAL L,SWX,COMP
      DIMENSION F(M),W(M),X(M),RLP(M),TP(M),X2(M)
      DIMENSION SI(K1),P(K1),CTP(11),CPS(11),
     1          CP(11),CLP(12)
C
C  INITIALIZE
C
      K=K1-1
      DO 10 I=1,K1
10    CP(I)=0.0E 00
      SIMIN=0.0E 00
      SWX=.TRUE.
      OM=0.0E 00
      B0=0.0E 00
      CLP(1)=0.0E 00
      CLP(2)=0.0E 00
      DELSQ=0.0E 00
      CTP(1)=0.10E 01
      TW=0.0E 00
      COMP=.FALSE.
      DO 20 I=1,M
      DELSQ=DELSQ+W(I)*F(I)*F(I)
      TP(I)=0.10E 01
      RLP(I)=0.0E 00
      OM=OM-W(I)*F(I)
20    TW=TW+W(I)
      S=OM/TW
      CP(1)=S
      DELSQ=DELSQ-S*OM
      SI(1)=DELSQ/FLOAT(M-1)
C*    SI(1)=DELSQ/DBLE(FLOAT(M-1))
      A=0.40E 01/(X(M)-X(1))
      B=-.20E 01-A*X(1)
      DO 30 I=1,M
30    X2(I)=A*X(I)+B
C  MAIN COMPUTATION LOOP
      DO 120 I=1,K
      DU=0.0E 00
      DO 40 J=1,M
40    DU=DU+W(J)*X2(J)*TP(J)*TP(J)
      AL=DU/TW
      RLW=TW
      OM=0.0E 00
      TW=0.0E 00
      DO 50 J=1,M
      DU=B0*RLP(J)
      RLP(J)=TP(J)
```

RF1LS AND AUX.


```
      TP(J)=(X2(J)-AL)*TP(J)-DU
      TW=TW+W(J)*TP(J)*TP(J)
50    OM=OM+W(J)*F(J)*TP(J)
      B1=TW/RLW
      S=OM/TW
      DELSQ=DELSQ-S*OM
      SI(I+1)=DELSQ/FLOAT(M-I-1)
C*    SI(I+1)=DELSQ/DBLE(FLOAT(M-I-1))
      IF(L) GO TO 90
      IF(COMP) GO TO 120
      IF(.NOT.SWX) GO TO 80
      IF(SI(I+1).LT.SI(I)) GO TO 70
C  HIGHER POWER APPEARS NOT TO IMPROVE FIT
      SWX=.FALSE.
      SIMIN=SI(I)
      DO 60 J=1,K1
60    CPS(J)=CP(J)
70    GO TO 90
80    IF(SI(I+1).LT.(0.6E 00*SIMIN)) COMP=.TRUE.
C
C  TERMINATION OF MAIN LOOP AT SUPERIOR FIT TO FIRST ONE FOUND.
C  RECURSION TO OBTAIN THE COEFFICIENTS CP OF THE POLYNOMIAL
C  OF BEST FIT OF DEGREE I+1.
C
90    DO 100 J=1,I
      DU=CLP(J+1)*B0
      CLP(J+1)=CTP(J)
      CTP(J)=CLP(J)-AL*CTP(J)-DU
100   CP(J)=CP(J)+S*CTP(J)
      CP(I+1)=S
      CTP(I+1)=0.10E 01
      CLP(I+2)=0.0E 00
      IF(COMP.OR.SWX) GO TO 120
      IF(I.NE.K) GO TO 120
      DO 110 J=1,K1
110   CP(J)=CPS(J)
120   B0=B1
      IF(L) GO TO 140
      DO 130 I=1,K
      IF(CP(K1).NE.0.0E 00) GO TO 140
130   K1=K1-1
140   CALL RF2LS(A,B,CP,P,K1,TP,RLP)
      RETURN
      END


      SUBROUTINE RF2LS(A,B,C,D,N1,Z,W)
C.    DOUBLE PRECISION A,B,C,D,Z,W
      DIMENSION C(N1),D(N1),Z(N1),W(N1)
      W(1)=0.10E 01
```

RF1LS AND AUX. Cont.


```
      Z(1)=0.10E 01
      D(1)=C(1)
      DO 10 I=2,N1
      W(I)=0.10E 01
      Z(I)=B*Z(I-1)
   10 D(1)=D(1)+C(I)*Z(I)
      DO 30 J=2,N1
      W(1)=W(1)*A
      D(J)=C(J)*W(1)
      K=2
      J1=J+1
      IF(J1.GT.N1) GO TO 30
      DO 20 I=J1,N1
      W(K)=A*W(K)+W(K-1)
      D(J)=D(J)+C(I)*W(K)*Z(K)
   20 K=K+1
   30 CONTINUE
      RETURN
      END
```

APPENDIX G

SPECIAL FUNCTIONS AND MACHINE CONSTANTS

The functions in this appendix are used to approximate some special functions and to determine machine constants. Table G summarizes the actions of the functions.

FUNCTION NAME	PURPOSE
RF1SF	Evaluates Sinh(x)
DF1SF	Double precision version of RF1SF
RF2SF	Evaluates Cosh(x)
DF2SF	Double precision version of RF2SF
RF3SF	Evaluates the error function Erf(x)
DF3SF	Double precision version of RF3SF
RF4SF	Evaluates Bessel function J_o
DF4SF	Double precision version of RF4SF
RF5SF	Evaluates Bessel function J_1
DF5SF	Double precision version of RF5SF
RF6SF	Evaluates Bessel function Y_o
DF6SF	Double precision version of RF6SF
RF7SF	Evaluates Bessel function Y_1
DF7SF	Double precision version of RF7SF
RF8SF	Evaluates $\sum' a_r T_r(t)$
DF8SF	Double precision version of RF8SF
IF1MQ	Determines integer machine dependent constants
RF2MQ	Determines machine relative error
DF2MQ	Double precision version of RF2MQ

TABLE G: Summary of functions.

1. FUNCTION RF1SF

a) Purpose

RF1SF approximates the function Sinh(x).

b) Specification

The function is written in FORTRAN and it is of the form:

REAL FUNCTION RF1SF (X)

c) Description

Sinh(x) is defined as:

$\text{Sinh}(x) = 0.5(e^{x}-e^{-x})$

but because of round off errors, function RF1SF uses the definition:

$\text{Sinh}(x) = 0.5\,(e^{x}-e^{-x}) \qquad x > 0.5$

$\qquad\quad = xF_n(x^2) \qquad\qquad x \leq 0.5$

Approximating polynomial is in Hart et al (81)

d) Parameters

The result is returned via the function name RF1SF. The only variable in the parameter list is X and it is not altered on exit. It is also real and it is the point at which Sinh is evaluated.

2 FUNCTION RF2SF

a) Purpose

RF2SF approximates the function Cosh(x)

b) Specification

The function is written in FORTRAN and it is of the form:

REAL FUNCTION RF2SF(X)

c) Description

Cosh(x) is defined as:

$$\mathrm{Cosh}(x) = 0.5(e^{x}+e^{-x})$$

and function RF2SF uses this formula.

d) Parameters

The result is returned via the function name RF2SF. The only variable in the parameter list is X and it is not altered on exit. It is also real and it is the point at which Cosh is evaluated.

3. FUNCTION RF3SF

a) PURPOSE

RF3SF is used to approximate the error function Erf(x)

b) Specification

The function is written in FORTRAN and it is of the form:

REAL FUNCTION RF3SF(X)

c) Description

The error function is defined as:

$$\mathrm{Erf}(x) = \frac{2}{\sqrt{\pi}}\int_{0}^{x} e^{-t^{2}}\,dt$$

and it is an increasing function of x. Function RF3SF approximates Erf(x) by the use of Chebyshev polynomials. (see Clenshaw (28) for approximating polynomial).

d) Parameters

The result is returned via the function name RF3SF. The only variable in the parameter list is X, and it is not altered on exit. It is also real and it is the point at which Erf is evaluated.

4. FUNCTION RF4SF

a) Purpose

RF4SF approximates the Bessel function J_o of the first kind.

b) Specification

The function is written in FORTRAN and it is of the form:

REAL FUNCTION RF4SF(X)

c) Description

Bessel functions of order ν are solutions of the differential equation:

$$z^2 \frac{d^2w}{dz^2} + z\frac{dw}{dz} + (z^2 - \nu^2)\, w = 0 \qquad 4a$$

For J_o, the order is zero. Function RF4SF approximates J_o by the use of Chebyshev polynomials. (see Clenshaw (28) for approximating Chebyshev polynomial).

d) Parameters

The result is returned via the function name RF4SF and the real variable X is not altered on exit. It is the point at which J_o is evaluated.

5. FUNCTION RF5SF

a) Purpose

RF5SF approximates the Bessel function J_1 of the first kind.

b) Specification

The function is written in FORTRAN and it is of the form:

REAL FUNCTION RF5SF(X)

c) Description

J_1 is a solution of equation 4a of order 1. Function RF5SF approximates J_1 by the use of Chebyshev polynomials (see Clenshaw (28) for approximating Chebyshev polynomial).

d) Parameters

The result is returned via the function name RF5SF and the real variable X is unaltered on exit. Also X is the point at which J_1 is evaluated.

6. FUNCTION RF6SF

a) Purpose

Function RF6SF is used to approximate the Bessel function Y_o of the second kind.

b) Specification

The function is written in FORTRAN and it is of the form:

```
REAL FUNCTION RF6SF(X,IFAIL)
```

c) Description

Y_o is also a solution of equation 4a, but it has logarithmatic branch points at the origin and it is not defined for negative or zero values of X. Function RF6SF approximates Y_o by the use of Chebyshev polynomials. (see Clenshaw (28) for approximating polynomial).

d) Parameters

The result is returned via the function name RF6SF and the real variable X is unaltered on exit. Also X is the point at which Y_o is evaluated. IFAIL is the error flag and it is an integer variable set to zero on successful exit otherwise non zero value.

e) Error diagnostics

On successful exit IFAIL is set to zero.

IFAIL	ERROR MESSAGE	RECOVERY
1	-VE OR ZERO VALUES NOT ACCEPTED BY RF6SF	Check X and change it to positive non-zero value

FUNCTION RF7SF

a) Purpose

Function RF7SF is used to approximate the Bessel Function Y_1 of the second kind.

b) Specification

The function is written in FORTRAN and it is of the form:

REAL FUNCTION RF7SF (X, IFAIL)

c) Description

Y_1 is the first order form of Y_o and Function RF7SF approximates Y_1 by the use of Chebyshev polynomial. (see Clenshaw (28) for approximating polynomial).

d) Parameters

The result is returned via the function name RF7SF and the real variable X is not altered on exit. Also X is the point where Y_1 is evaluated and must be specified by the user before entry to the function. IFAIL is an integer and it is the error flag set to zero on successful exit, otherwise non-zero value.

e) Error diagnostics

IFAIL	ERROR MESSAGE	RECOVERY
1	-VE OR ZERO VALUES NOT ACCEPTED BY RF7SF	Check X and change it to positive, non-zero value.

8. FUNCTION RF85F

a) Purpose

RF8SF evaluates a Chebyshev polynomial at a given point.

b) Specification

The function is written in FORTRAN and it is of the form:

```
REAL FUNCTION RF8SF (A,T,N)
```

c) Description

RF8SF evaluates $f(x) = \sum_{r=0}^{n}{}' a_r T_r(x)$

by using the recurrence relation

$$b_r = 2x b_{r+1} - b_{r+2} + a_r \quad , \quad b_{n+1} = b_{n+2} = 0$$

$$r = n, n-1, \ldots, 0$$

and finally obtaining f(x) by

$$f(x) = \frac{1}{2}(b_o - b_2)$$

d) Parameters

A	real one-dimensional array of size N. On entry A specifies the coefficients of the Chebyshev polynomial. Not altered on exit.
T	real variable which on entry, the user must set to the point where the polynomial is to be evaluated. Not altered on exit.
N	integer variable which on entry the user should set to the number of coefficients of Chebyshev polynomial to be evaluated. Not altered on exit.

RF8SF Function name which on exit contains the result of the evaluation.

NOTE: RF8SF (and DF8SF if double precision arithmetic is available) is in file BASE which means that it is not linked sperately like others.

9. FUNCTION IF1MQ

a) Purpose

This function is used to obtain integer machine dependent constants.

b) Specification

The function is written in FORTRAN and it is of the form:

```
INTEGER FUNCTION IF1MQ(I)
```

c) Description

The value of I must be such that $1 \leq I \leq 8$ if double precision arithmetic is available and $1 \leq I \leq 7$ if only single precision arithmetic is available. Each value of I specifies a particular integer machine dependent quantity to be computed.

I	MACHINE QUANTITY
1	Standard output unit number of the system.
2	Base of the arithmetic used by the machine.
3	Number of base-B digits in the mantissa of a single precision floating-point number.
4	Maximum exponent (Emax) of floating point numbers such that $\lvert x \rvert < 10^{Emax}$
5	Minimum exponent (Emin) of floating-point numbers such that $10^{Emin} < \lvert x \rvert$
6	Maximum integer representable
7	Minimum integer representable
8	Number of base-B digits in the mantissa of a double-precision floating-point number if double precision arithmetic is provided.

d) Parameters

The result is returned via the function name IF1MQ and the integer variable I which specifies what machine dependent quantity is to be determined. It is not altered on exit and it should be specified on entry by the user.

NOTE: IF1MQ is in the file BASE.

10. FUNCTION RF2MQ

a) Purpose

This function computes the machine relative error.

b) Specification

RF2MQ is written in FORTRAN and it is of the form:

```
REAL FUNCTION RF2MQ(R)
```

c) Description

RF2MQ obtains the machine relative error TOL defined as:

$$TOL = \begin{cases} \beta^{1-t} & \text{Chopped arithmetic} \\ \frac{1}{2}\beta^{1-t} & \text{rounded arithmetic} \end{cases}$$

where β is the base of the arithmetic used and t the mantissa length. length.

d) Parameters

The result is returned via the function name RF2MQ. The variable R has nothing to do with the computation. It is there so that RF2MQ can satisfy the definition of a Function in FORTRAN. R can therefore be given any value on entry and it is not altered on exit.

NOTE: RF2MQ (and DF2MQ if double precision arithmetic is available) is in file BASE.

10. DOUBLE PRECISION VERSIONS

The functions DF1SF, DF2SF, DF3SF, DF4SF DF5SF, DF6SF, DF7SF DF8SF and DF2MQ are the double precision versions of functions RF1SF, FR2SF RF3SF, RF4SF, RF5SF, RF6SF RF7SF, RF8SF, and RF2MQ

respectively. When these double precision versions are in use, they should be declared as DOUBLE PRECISION together with the real variables in their parameter list in the (sub) program in which they are referenced. The results produced are usually more accurate than those of single precision versions.

These DOUBLE PRECISION versions are only implemented if double precision arithmetic is available.

FUNCTION RF1SF

```
      REAL FUNCTION RF1SF(X)
C  RF1SF COMPUTES THE SINH OF X IN SINGLE PRECISION
C  USING POLYNOMIAL APPROXIMATION OR 0.5*(EXP(X)-EXP(-X))
      REAL X,Y,SUM
      IF(ABS(X) .LE. 0.5000000E 00) GO TO 10
      Y=EXP(X)
      RF1SF=0.50000E 00*(Y-(0.100E 01/Y))
      RETURN
10    Y=X*X
      SUM=Y*0.19996829E-03+0.83330413E-02
      SUM=SUM*Y+0.16666669E 00
      SUM=SUM*Y+0.10000000E 01
      RF1SF=X*SUM
      RETURN
      END
```

FUNCTION DF1SF

```
      DOUBLE PRECISION FUNCTION DF1SF(X)
C  DF1SF COMPUTES THE SINH OF X IN DOUBLE PRECISION USING
C  EITHER POLYNOMIAL APPROXIMATION OR 0.5(EXP(X)-EXP(-X))
      DOUBLE PRECISION X,Y,SUM
      IF(DABS(X) .LE. 0.500000D 00) GO TO 10
      Y=DEXP(X)
      DF1SF=(Y-(0.10000D 01/Y))*0.5000000D 00
      RETURN
10    Y=X*X
      SUM=0.2518290665924D-07*Y+0.2755691013580 94D-05
      SUM=SUM*Y+0.198412704551762D-03
      SUM=SUM*Y+0.833333333288556D-02
      SUM=SUM*Y+0.166666666666668D 00
      SUM=SUM*Y+0.100000000000000D 01
      DF1SF=X*SUM
      RETURN
      END
```

FUNCTION RF2SF

```
      REAL FUNCTION RF2SF(X)
C  RF2SF COMPUTES THE COSH OF X IN SINGLE PRECISION
C  USING THE FORMULA 0.5*(EXP(X)+EXP(-X))
      REAL X,Y
      Y=EXP(X)
      RF2SF=(Y+(0.1000000E 01/Y))*0.5000000E 00
      RETURN
      END
```

FUNCTION DF2SF

```
      DOUBLE PRECISION FUNCTION DF2SF(X)
C  DF2SF EVALUATES THE COSH OF X IN DOUBLE PRECISION
C  USING THE FORMULA 0.5*(EXP(X)+EXP(-X))
      DOUBLE PRECISION X,Y
      Y=DEXP(X)
      DF2SF=(Y+(0.10000000D 01/Y))*0.5000000D 00
      RETURN
      END
```

FUNCTION RF3SF

```
      REAL FUNCTION RF3SF(X)
C  RF3SF EVALUATES THE ERROR FUNCTION ERF AT A GIVEN
C  POINT X IN SINGLE PRECISION USING CHEBYSHEV POLYNOMIALS
      REAL A,X,T,XT,XT2
      DIMENSION A(25)
      DATA A(1),A(2),A(3)/
     1     0.38873037E 01,-0.13816314E 01,0.64731640E 00/
      DATA A(4),A(5),A(6)/
     1     -0.30593102E 00,0.13867975E 00,-0.59247457E-01/
      DATA A(7),A(8),A(9)/
     1     0.23691752E-01,-0.88473626E-02,0.30856617E-02/
      DATA A(10),A(11),A(12)/
     1     -0.10063863E-02,0.30754633E-03,-0.88261984E-04/
      DATA A(13),A(14),A(15)/
     1     0.23845096E-04,-0.60791003E-05,0.14659722E-05/
      DATA A(16),A(17),A(18)/
     1     -0.33515993E-06,0.72805795E-07,-0.15057912E-07/
      DATA A(19)/0.29709474E-08/
      DATA A(20),A(21),A(22)/
     1     -0.56021274E-09,0.10114162E-09,-0.17506505E-10/
      DATA A(23),A(24),A(25)/
     1     0.29103814E-11,-0.46532645E-12,0.7164815E-13/
C
      RF3SF=0.0
      IF(X.EQ.0.0) RETURN
      RF3SF=0.10E 01
      IF(ABS(X).LE.0.40E 01) GO TO 10
      IF(X.LT.0.0E 00) RF3SF=-0.10E 01
      RETURN
10    T=0.25E 00*X
      XT=0.20E 01*T*T-0.100E 01
      RF3SF=T*RF8SF(A,XT,25)
      RETURN
      END
```

FUNCTION DF3SF

```
      DOUBLE PRECISION FUNCTION DF3SF(X)
C  THIS ROUTINE COMPUTES THE ERROR FUNCTION
C  IN DOUBLE PRECISION USING CHEBYSHEV POLYNOMIALS
      DOUBLE PRECISION A,B,X,T,T2,XT,XT2,DF8SF
      DIMENSION A(31),B(16)
      DATA A(1),A(2),A(3),A(4)/
     1     0.3887303655222904D 01,-0.1381631420019799D 01,
     2     0.6473164048545842D 00,-0.3059310244220356D 00/
      DATA A(5),A(6),A(7),A(8),A(9),A(10)/
     1     0.1386797472020001D 00,-0.5924745659125903D-01,
     2     0.2369175182492819D-01,-0.8847362635240453D-02,
     3     0.3085661711360924D-02,-0.1006386351237978D-02/
      DATA A(11),A(12),A(13),A(14),A(15),A(16)/
     1     0.3075463288430778D-03,-0.8826198375536312D-04,
     2     0.2384509616607263D-04,-0.607910028505827D-05,
     3     0.146597217338083D-05,-0.33515993427206D-06/
      DATA A(17),A(18),A(19),A(20),A(21),A(22)/
     1     0.7280579544232D-07,-0.1505791176668D-07,
     2     0.297094742055D-08,-0.56021273938D-09,
     3     0.10113162390D-09,-0.1750650485D-10/
      DATA A(23),A(24),A(25),A(26),A(27),A(28)/
     1     0.291033139D-11,-0.46532645D-12,
     2     0.7164815D-13,-0.1063749D-13,
     3     0.152467D-14,-0.21120D-15/
      DATA A(29),A(30),A(31)/
     1     0.283100D-16,-0.3670D-17,
     2     0.460D-18/
C
      DATA B(1),B(2),B(3),B(4),B(5),B(6)/
     1     0.1970705272257545D 01,-0.1433974027177498D-01,
     2     0.2973616922026189D-03,-0.9803516043336237D-05,
     3     0.433133420347 28D-06,-0.2362150026241D-07/
      DATA B(7),B(8),B(9),B(10),B(11),B(12)/
     1     0.151549676581D-08,-0.110034939856D-09,
     2     0.904259014D-11,-0.80947054D-12,
     3     0.7853856D-13,0.817918D-14/
      DATA B(13),B(14),B(15),B(16)/
     1     0.90715D-15,-0.10646D-15,
     2     0.1315D-16,-0.170D-17/
C
      DF3SF=0.0D 00
      T=DABS(X)
      IF(T.EQ.0.0D 00) RETURN
      DF3SF=0.10D 01
      IF(T.LE.0.650D 01) GO TO 10
      IF(X.LT.0.0D 00) DF3SF=-0.10D 01
      RETURN
10    IF(T.GT. 0.400D 01) GO TO 20
      XT=0.25D 00*X
      T2=0.20D 01*XT*XT-0.100D 01
      DF3SF=XT*DF8SF(A,T2,31)
      RETURN
20    XT=0.40D 01/T
      T2=0.20D 01*XT*XT-0.100D 01
      XT2=DF8SF(B,T2,16)
```

DF3SF Cont.

```
      XT=DEXP(-T*T)*0.5641895835477563D 00/T
      XT2=0.100D 01-XT*XT2
      IF(X.LT.0.0D 00) XT2=-XT2
      DF3SF=XT2
      RETURN
      END
```

FUNCTION RF4SF

```
      REAL FUNCTION RF4SF(X)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION J0 AT X
C  IN SINGLE PRECISION USING CHEBYSHEV POLYNOMIALS
      REAL A,P,Q,X,S1,S2,T,T2,T3,EP
      DIMENSION A(14),P(8),Q(8)
      DATA A(1),A(2),A(3),A(4)/
     1   0.31545594E 00,-0.87234423E-02,0.26517861E 00,-0.37009499E 00/
      DATA A(5),A(6),A(7),A(8)/
     1 0.15806710E 00,-0.34893769E-01,0.48191801E-02,-0.46062617E-03/
      DATA A(9),A(10),A(11),A(12)/
     1 0.32460329E-04,-0.17619469E-05,0.76081636E-07,-0.26792535E-08/
      DATA A(13),A(14)/
     1 0.78486963E-10,-0.19438347E-11/
C
      DATA P(1),P(2),P(3),P(4),P(5),P(6),P(7),P(8)/
     1 0.19989207E 01,-0.53452205E-03,0.30751848E-05,-0.51705945E-07,
     2 0.16306465E-08,-0.78640914E-10,0.51682624E-11,-0.43045789E-12/
C
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8)/
     1 -0.31111709E-01,0.68385199E-04,-0.74144984E-06,0.17972457E-07,
     2 -0.72719159E-09,0.42201219E-10,-0.32067474E-11,0.30061451E-12/
C
      RF4SF=0.100E 01
      T=ABS(X)
      T2=RF2MQ(0.0)
      T2=SQRT(T2)
      IF(T.LT.T2) RETURN
      IF(T.GT.0.80E 01) GO TO 20
      T=0.125E 00*T
      T=0.20E 01*T*T-0.10E 01
      RF4SF=RF8SF(A,T,14)
      RETURN
20    EP=T-0.78539816E 00
      T2=SQRT(T)
      T=0.80E 01/T
      T3=0.200E 01*T*T-0.100E 01
      S1=RF8SF(P,T3,8)
      S2=RF8SF(Q,T3,8)
      RF4SF=0.79788456E 00*(S1*COS(EP)-S2*SIN(EP))/T2
      RETURN
      END
```

FUNCTION DF4SF

```
      DOUBLE PRECISION FUNCTION DF4SF(X)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION J0 AT POINT X
C  IN DOUBLE PRECISION
      DOUBLE PRECISION A,P,Q,X,S1,S2,T,T2,T3,EP,DF2MQ,DF8SF
      DIMENSION A(17),P(14),Q(14)
      DATA A(1),A(2),A(3),A(4),A(5),A(6)/
     1     0.315455942949780 2D 00,-0.8723442352852221D-02,
     2     0.2651786132033368D 00,-0.3700949938726498D 00,
     3     0.1580671023320973D 00,-0.3489376941140868D-01/
      DATA A(7),A(8),A(9),A(10),A(11),A(12)/
     1     0.4819180069467604D-02,-0.4606261662062750D-03,
     2     0.3246032882100508D-04,-0.1761946907762150D-05,
     3     0.7608163592419D-07,-0.267925353056D-08/
      DATA A(13),A(14),A(15),A(16),A(17)/
     1     0.7848696314D-10,-0.194333469D-11,
     2     0.4125321D-13,-0.75885D-15,0.1222D-16/
C
      DATA P(1),P(2),P(3),P(4),P(5),P(6)/
     1     0.1998920698695037 3D 01,-0.5365220468132117D-03,
     2     0.3075184787 5_947D-05,-0.5170594537606D-07,
     3     0.163064646352D-08,-0.7864091377D-10/
      DATA P(7),P(8),P(9),P(10),P(11),P(12),P(13),P(14)/
     1     0.516826239D-11,-0.43045789D-12,0.4326596D-13,
     2     -0.506903D-14,0.67481D-15,-0.10012D-15,0.1631D-16,
     3     -0.288D-17/
C
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6)/
     1     -0.3111170921067 40D-01,0.6838519942611650D-04,
     2     -0.7414498411060 6D-06,0.1797245724797D-07,
     3     -0.72719159369D-09,0.4220121905D-10/
      DATA Q(7),Q(8),Q(9),Q(10),Q(11),Q(12),Q(13),Q(14)/
     1     -0.320674742D-11,0.30061451D-12,-0.3336328D-13,
     2  0.425523D-14,-0.60999D-15,0.9662D-16,-0.1669D-16,0.311D-17/
C
      DF4SF=0.100D 01
      T=DABS(X)
      T2=DF2MQ(0.0D 00)
      T2=DSQRT(T2)
      IF(T.LT.T2) RETURN
      IF(T.GT.0.80D 01) GO TO 20
      T=0.125D 00*T
      T=0.20D 01*T*T-0.10E 01
      DF4SF=DF8SF(A,T,17)
      RETURN
20    EP=T-0.7853981633974483D 00
      T2=DSQRT(T)
      T=0.80D 01/T
      T3=0.200D 01*T*T-0.100D 01
      S1=DF8SF(P,T3,14)
      S2=T*DF8SF(Q,T3,14)
      DF4SF=0.7978845608028654D 00*(S1*DCOS(EP)-S2*DSIN(EP))/T2
      RETURN
      END
```

FUNCTION RF5SF

```
      REAL FUNCTION RF5SF(X)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION J1
C  IN SINGLE PRECISION USING CHEBYSHEV POLYNOMIALS
      REAL A,P,Q,X,S1,S2,T,T2,XT2,T3,EP
      DIMENSION A(12),P(5),Q(5)
      DATA A(1),A(2),A(3),A(4),A(5),A(6)/
     1     0.12967175E 01,-0.11918012E 01, 0.12879941E 01,
     2    -0.66144393E 00, 0.17770912E 00,-0.29175525E-01/
      DATA A(7),A(8),A(9),A(10),A(11),A(12)/
     1     0.32402702E-02,-0.26044439E-03, 0.15887019E-04,
     2    -0.76175878E-06, 0.29497070E-07,-0.94242130E-09/
C
      DATA P(1),P(2),P(3),P(4),P(5)/
     1     0.20018061E 01, 0.89898983E-03,-0.39872843E-05,
     2     0.61776340E-07,-0.18718907E-08/
      DATA Q(1),Q(2),Q(3),Q(4),Q(5)/
     1     0.93555574E-01,-0.96277235E-04,0.91386153E-06,
     2    -0.20959781E-07, 0.82291933E-09/
C
      T=ABS(X)
      IF(T.GT.0.80E 01) GO TO 20
      IF(T.GT.0.200E-02) GO TO 5
      RF5SF=0.5E 00*X
      RETURN
5     T=0.125E 00*T
      T2=0.20E 01*T*T-0.10E 01
      S1=T*RF8SF(A,T2,12)
      IF(X .LT. 0.0E 00) S1=-S1
      RF5SF=S1
      RETURN
20    EP=T-0.23561945E 01
      T2=SQRT(T)
      T=0.80E 01/T
      T3=0.200E 01*T*T-0.100E 01
      S1=RF8SF(P,T3,5)
      S2=T*RF8SF(Q,T3,5)
      S1=0.79788456E 00*(S1*COS(EP)-S2*SIN(EP))/T2
      IF(X .LT. 0.0E 00) S1=-S1
      RF5SF=S1
      RETURN
      END
```

FUNCTION DF5SF

```
      DOUBLE PRECISION FUNCTION DF5SF(X)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION J1
C  IN DOUBLE PRECISION USING CHEBYSHEV POLYNOMIALS
      DOUBLE PRECISION A,P,Q,X,S1,S2,T,T2,T3,EP,DF2MQ,DF8SF
      DIMENSION A(17),P(14),Q(14)
      DATA A(1),A(2),A(3),A(4),A(5),A(6)/
     1    0.1296717541210530D 01,-0.1191801160541217D 01,
     2    0.1287994098857678D 01,-0.6614439341345432D 00,
     3    0.1777091172397283D 00,-0.2917552480615421D-01/
      DATA A(7),A(8),A(9),A(10),A(11),A(12)/
     1    0.3240270182683857D-02,-0.2604443893485807D-03,
     2    0.1568701923993213D-04,-0.7617587805400 3D-06,
     3    0.2949707007278D-07,-0.94242129816D-09/
      DATA A(13),A(14),A(15),A(16),A(17)/
     1    0.2528123664D-10,-0.57774042D-12,
     2    0.1138572D-13,-0.19554D-15, 0.295D-17/
C
      DATA P(1),P(2),P(3),P(4),P(5),P(6)/
     1    0.2001806081720027D 01, 0.8989898330859408D-03,
     2   -0.3987284300048891D-05, 0.6177633960644D-07,
     3   -0.187189074911D-08, 0.8816898660D-10/
      DATA P(7),P(8),P(9),P(10),P(11),P(12),P(13),P(14)/
     1   -0.5704863264D-11, 0.46991955D-12,
     2   -0.4684224D-13, 0.545267D-14,-0.72212D-15,
     3    0.10668D-15,-0.1731D-16, 0.305D-17/
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6)/
     1    0.9355557413907065D-01,-0.9627723549157079D-04,
     2    0.9138615257955 5D-06,-0.2095978138408D-07,
     3    0.82291933277D-09,-0.4686363688D-10/
      DATA Q(7),Q(8),Q(9),Q(10),Q(11),Q(12),Q(13),Q(14)/
     1    0.351521879D-11,-0.32643157D-12,
     2    0.3596777D-13,-0.456125D-14, 0.65083D-15,
     3   -0.10269D-15, 0.1768D-16,-0.328D-17/
C
      T=DABS(X)
      IF(T.GT.0.8CD 01) GO TO 20
      T2=DF2MQ(0.0)
      T2=0.1D 00*DSQRT(T2)
      IF(T.GT.T2) GO TO 5
      DF5SF=0.5D 00*X
      RETURN
5     T=0.125D 00*T
      T2=0.20D 01*T*T-0.10D 01
      S1=T*DF8SF(A,T2,17)
      IF(X .LT. 0.0D 00) S1=-S1
      DF5SF=S1
      RETURN
20    EP=T-0.2356194490192345D 01
      T2=DSQRT(T)
      T=0.80D 01/T
      T3=0.200D 01*T*T-0.100D 01
      S1=DF8SF(P,T3,14)
      S2=T*DF8SF(Q,T3,14)
      S1=0.7978845608028654D 00*(S1*DCOS(EP)-S2*DSIN(EP))/T2
      IF(X .LT. 0.0D 00) S1=-S1
```

DF5SF Cont.

```
0056        DF5SF=S1
0057        RETURN
0058        END
```

FUNCTION RF6SF

```
      REAL FUNCTION RF6SF(X,IFAIL)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION Y0
C  IN SINGLE PRECISION USING CHEBYSHEV POLYNOMIALS
      REAL A,P,Q,X,S1,S2,T,T2,XT2,T3,EP,RF4S
      DIMENSION A(14),Y(13),P(8),Q(8)
      DATA A(1),A(2),A(3),A(4)/
     1   0.31545594E 00,-0.87234423E-02,0.26517861E 00,-0.37009499E 00/
      DATA A(5),A(6),A(7),A(8)/
     1 0.15806710E 00,-0.34893769E-01,0.48191801E-02,-0.46062617E-03/
      DATA A(9),A(10),A(11),A(12)/
     1 0.32460329E-04,-0.17619469E-05,0.76081636E-07,-0.26792535E-08/
      DATA A(13),A(14)/
     1 0.78486963E-10,-0.19438347E-11/
C
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)/
     1     -0.66292226E-01,-0.27447430E 00,
     2      0.17903431E 00, 0.26156725E 00,
     3     -0.17730201E 00, 0.47196690E-01/
      DATA Y(7),Y(8),Y(9),Y(10),Y(11),Y(12),Y(13)/
     1     -0.72879625E-02, 0.75311359E-03,
     2     -0.56320791E-04, 0.32065325E-05,
     3     -0.14407233E-06, 0.52487948E-07,-0.15837552E-09/
C
      DATA P(1),P(2),P(3),P(4),P(5),P(6),P(7),P(8)/
     1 0.19989207E 01,-0.53652205E-03,0.30751848E-05,-0.51705945E-07,
     2 0.16306465E-08,-0.78640914E-10,0.51682624E-11,-0.43045789E-12/
C
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8)/
     1 -0.31111709E-01,0.68385199E-04,-0.74144984E-05,0.17972457E-07,
     2 -0.72719159E-09,0.42201219E-10,-0.32067474E-11,0.3006_451E-12/
C
      IF(X.GT.0.0) GO TO 5
      IOUT=IF1MQ(1)
      IFAIL=1
      WRITE(IOUT,100) IFAIL
      RETURN
100   FORMAT(I8,4X,40H-VE OR ZERO VALUES NOT ACCEPTED BY RF6SF/)
5     IFAIL=0
      T=X
      IF(T.GT.0.80E 01) GO TO 20
      EP=RF2MQ(0.0)
      EP=SQRT(EP)
      IF(T.GT.EP) GO TO 10
      RF6SF=0.63661977E 00*(ALOG(0.50E 00*X)+0.57721566E 00)
      RETURN
10    T=0.125E 00*T
      T=0.20E 01*T*T-0.10E 01
      RF4S=RF8SF(A,T,14)
      T2=RF8SF(Y,T,13)
      RF6SF=0.63661977E 00*RF4S*ALOG(X)+T2
      RETURN
20    EP=T-0.78539816E 00
      T2=SQRT(T)
      T=0.80E 01/T
      T3=0.200E 01*T*T-0.100E 01
```

RF6SF Cont.

```
      S1=RF8SF(P,T3,8)
      S2=T*RF8SF(Q,T3,8)
      RF6SF=0.79788456E 00*(S2*COS(EP)+S1*SIN(EP))/T2
      RETURN
      END
```

FUNCTION DF6SF

```
      DOUBLE PRECISION FUNCTION DF6SF(X,IFAIL)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUN Y0 IN DOUBLE PRECISION
C  USING CHEBYSHEV POLYNOMIALS
      DOUBLE PRECISION A,Y,P,Q,X,S1,S2,T,T2,T3,EP,DF2MQ,DF8SF
      DIMENSION A(17),Y(17),P(14),Q(14)
      DATA A(1),A(2),A(3),A(4),A(5),A(6)/
     1     0.3154559429497802D 00,-0.8723442352852221D-02,
     2     0.2651786132033368D 00,-0.3700949938726498D 00,
     3     0.1580671023320973D 00,-0.3489376941140868D-01/
      DATA A(7),A(8),A(9),A(10),A(11),A(12)/
     1     0.4819180069467604D-02,-0.4606261662062750D-03,
     2     0.3246032882100508D-04,-0.1761946907762150D-05,
     3     0.7608163592419D-07,-0.267925353056D-08/
      DATA A(13),A(14),A(15),A(16),A(17)/
     1     0.7848696314D-10,-0.194383469D-11,
     2     0.4125321D-13,-0.75885D-15,0.1222D-16/
C
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)/
     1     -0.6629222640656988D-01,-0.2744743055297453D 00,
     2      0.1790343140771827D 00, 0.2615673462550466D 00,
     3     -0.1773020127811436D 00, 0.4719668959576339D-01/
      DATA Y(7),Y(8),Y(9),Y(10),Y(11),Y(12)/
     1     -0.7287962479552079D-02, 0.7531135932577742D-03,
     2     -0.5632079141056987D-04, 0.3206532537654480D-05,
     3     -0.1440723327401 9D-06, 0.524879478733D-08/
      DATA Y(13),Y(14),Y(15),Y(16),Y(17)/
     1     -0.15837552542D-09, 0.402633082D-11,
     2     -0.8747341D-13, 0.164349D-14,-0.2698D-16/
C
      DATA P(1),P(2),P(3),P(4),P(5),P(6)/
     1     0.1998920698695037 3D 01,-0.5365220468132117D-03,
     2     0.307518478751947D-05,-0.5170594537606D-07,
     3     0.163064646352D-08,-0.7864091377D-10/
      DATA P(7),P(8),P(9),P(10),P(11),P(12),P(13),P(14)/
     1     0.516826239D-11,-0.43045789D-12,0.4326596D-13,
     2     -0.506903D-14,0.67481D-15,-0.10012D-15,0.1631D-16,
     3     -0.288D-17/
C
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6)/
     1     -0.311117092106740D-01,0.6838519942611650D-04,
     2     -0.74144984110606D-06,0.1797245724797D-07,
     3     -0.72719159369D-09,0.4220121905D-10/
      DATA Q(7),Q(8),Q(9),Q(10),Q(11),Q(12),Q(13),Q(14)/
     1     -0.320674742D-11,0.30061451D-12,-0.3336328D-13,
     2  0.425523D-14,-0.60999D-15,0.9662D-16,-0.1669D-16,0.311D-17/
C
      IF(X.GT.0.0D 00) GO TO 5
      IFAIL=1
      IOUT=IF1MQ(1)
      WRITE(IOUT,100) IFAIL
100   FORMAT(I8,4X,40H-VE OR ZERO VALUES NOT ACCEPTED BY DF6SF/)
      RETURN
5     IFAIL=0
      T=X
      IF(T.GT.0.90D 01) GO TO 20
```

DF6SF Cont.

```
      T3=DF2MQ(0,0)
      T3=DSQRT(T3)
      IF(T.GT.T3) GO TO 10
      DF6SF=0.6366197723675813D 00*(DLOG(0.50D 00*T)
     1   +0.5772156649015329D 00)
      RETURN
10    T=0.125D 00*T
      T=0.20D 01*T*T-0.10D 01
      T3=DF8SF(A,T,17)
      S2=DF8SF(Y,T,17)
      DF6SF=0.6366197723675813D 00*T3*DLOG(X)+S2
      RETURN
20    EP=T-0.7853981633974483D 00
      T2=DSQRT(T)
      T=0.80D 01/T
      T3=0.200D 01*T*T-0.100D 01
      S1=DF8SF(P,T3,14)
      S2=T*DF8SF(Q,T3,14)
      DF6SF=0.7978845608028654D 00*(S2*DCOS(EP)+S1*DSIN(EP))/T2
      RETURN
      END
```

FUNCTION RF7SF

```
      REAL FUNCTION RF7SF(X,IFAIL)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION Y1
C  IN SINGLE PRECISION USING CHEBYSHEV POLYNOMIALS
      REAL A,P,Q,X,S1,S2,T,T2,XT2,T3,EP
      DIMENSION A(12),Y(12),P(5),Q(5)
      DATA A(1),A(2),A(3),A(4),A(5),A(6)/
     1     0.12967175E 01,-0.11918012E 01, 0.12679941E 01,
     2    -0.66144393E 00, 0.17770912E 00,-0.29175525E-01/
      DATA A(7),A(8),A(9),A(10),A(11),A(12)/
     1     0.32402702E-02,-0.26044439E-03, 0.15887019E-04,
     2    -0.76175878E-06, 0.29497070E-07,-0.94242130E-09/
C
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)/
     1     0.40608212E-01,-0.12869738E 00,-0.76729636E 00,
     2     0.67561578E 00,-0.22662499E 00, 0.42319180E-01/
      DATA Y(7),Y(8),Y(9),Y(10),Y(11),Y(12)/
     1    -0.51316412E-02, 0.44047863E-03,-0.28304640E-04,
     2     0.14166244E-05,-0.56884400E-07, 0.18754703E-08/
C
      DATA P(1),P(2),P(3),P(4),P(5)/
     1     0.20018061E 01, 0.89898983E-03,-0.39872843E-05,
     2     0.61776340E-07,-0.18718907E-08/
      DATA Q(1),Q(2),Q(3),Q(4),Q(5)/
     1     0.93555574E-01,-0.96277235E-04,0.91384153E-06,
     2    -0.20959781E-07, 0.82291933E-09/
C
      DATA CON/0.63661977E 00/
      IF(X.GT.0.0) GO TO 4
      IFAIL=1
      IOUT=IF1MQ(1)
      WRITE(IOUT,100) IFAIL
100   FORMAT(I8,4X,40H-VE OR ZERO VALUES NOT ACCEPTED BY RF7SF/)
      RETURN
4     IFAIL=0
      T=X
      IF(T.GT.0.80E 01) GO TO 30
      T3=RF2MQ(0.0)
      T3=0.10E 00*SQRT(T3)
      IF(T.GT.T3) GO TO 5
      RF7SF=-CON/T
      RETURN
5     T=0.125E 00*T
      T2=0.20E 01*T*T-0.10E 01
      IF(X.GT.0.200E-02) GO TO 10
      S1=0.5E 00*X
      GO TO 20
10    S1=T*RF8SF(A,T2,12)
20    S2=T*RF8SF(Y,T2,12)
      RF7SF=CON*ALOG(X)*S1-CON/X+S2
      RETURN
30    EP=T-0.23561945E 01
      T2=SQRT(T)
      T=0.80E 01/T
      T3=0.200E 01*T*T-0.100E 01
      S1=RF8SF(P,T3,5)
```

RF7SF Cont.

```
0056      S2=T▸RF8SF(G,T3,5)
0057      RF7SF=0.79788456E 00*(S2*COS(EP)+S1*SIN(EP))/T2
0058      RETURN
0059      END
```

FUNCTION DF7SF

```
      DOUBLE PRECISION FUNCTION DF7SF(X,IFAIL)
C  THIS FUNCTION APPROXIMATES THE BESSEL FUNCTION Y1
C  IN DOUBLE PRECISION USING CHEBYSHEV POLYNOMIALS
      DOUBLE PRECISION A,Y,P,Q,X,S1,S2,T2,CON,T3,T,EP,DF2MQ,DF8SF
      DIMENSION A(17),Y(17),P(14),Q(14)
      DATA A(1),A(2),A(3),A(4),A(5),A(6)/
     1    0.1296717541210530D 01,-0.1191801160541217D 01,
     2    0.1287994098857678D 01,-0.6614439341345432D 00,
     3    0.1777091172397283D 00,-0.2917552480615421D-01/
      DATA A(7),A(8),A(9),A(10),A(11),A(12)/
     1    0.3240270182683857D-02,-0.2604443893485807D-03,
     2    0.1588701923993213D-04,-0.7617587805400 3D-06,
     3    0.2949707007278D-07,-0.94242129816D-09/
      DATA A(13),A(14),A(15),A(16),A(17)/
     1    0.2528123664D-10,-0.57774042D-12,
     2    0.1138572D-13,-0.19554D-15, 0.295D-17/
C
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)/
     1     0.4060821177186851D-01,-0.1266973843813500D 00,
     2    -0.7672963628866459D 00, 0.6756157807721877D 00,
     3    -0.2266249915567549D 00, 0.4231918035333690D-01/
      DATA Y(7),Y(8),Y(9),Y(10),Y(11),Y(12)/
     1   -0.5131641161061085D-02, 0.4404786298670995D-03,
     2   -0.2830464014951480D-04, 0.1416524336449235D-05,
     3   -0.5688440039919D-07, 0.187547032473D-08/
      DATA Y(13),Y(14),Y(15),Y(16),Y(17)/
     1   -0.5172121473D-10, 0.121143321D-11,
     2   -0.2440949D-13, 0.42773D-15,-0.658D-17/
C
      DATA P(1),P(2),P(3),P(4),P(5),P(6)/
     1    0.2001806081720027D 01, 0.8989898330859408D-03,
     2   -0.398728430048891D-05, 0.617763396644D-07,
     3   -0.187189074911D-08, 0.8816898660D-10/
      DATA P(7),P(8),P(9),P(10),P(11),P(12),P(13),P(14)/
     1   -0.570486364D-11, 0.46991955D-12,
     2   -0.4684224D-13, 0.545267D-14,-0.72212D-15,
     3    0.10668D-15,-0.1731D-16, 0.305D-17/
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6)/
     1    0.9355557413907065D-01,-0.9627723549157079D-04,
     2    0.9138415257955D-06,-0.2095978138408D-07,
     3    0.82291933277D-09,-0.4686363688D-10/
      DATA Q(7),Q(8),Q(9),Q(10),Q(11),Q(12),Q(13),Q(14)/
     1    0.351521879D-11,-0.32643157D-12,
     2    0.3596777D-13,-0.456125D-14, 0.65083D-15,
     3   -0.10269D-15, 0.1768D-16,-0.328D-17/
C
      DATA CON/0.6366197723675813D 00/
      IF(X.GT.0.0D 00) GO TO 4
      IFAIL=1
      IOUT=IF1MQ(1)
      WRITE(IOUT,100) IFAIL
100   FORMAT(I8,4X,40H-VE OR ZERO VALUES NOT ACCEPTED BY DF7SF/)
      RETURN
4     IFAIL=0
      T=X
```

DF7SF Cont.

```
      IF(T.GT.0.80D 01) GO TO 20
      T2=DF2MQ(0.0D 00)
      T2=0.1D 00*DSQRT(T2)
      IF(T.GT.T2) GO TO 5
      DF7SF=-CON/T
      RETURN
5     T=0.125D 00*T
      T2=0.20D 01*T*T-0.10D 01
      S1=T*DF8SF(A,T2,17)
10    S2=T*DF8SF(Y,T2,17)
      DF7SF=CON*DLOG(X)*S1-CON/X+S2
      RETURN
20    EP=T-0.2356194490192345D 01
      T2=DSQRT(T)
      T=0.80D 01/T
      T3=0.200D 01*T*T-0.100D 01
      S1=DF8SF(P,T3,14)
      S2=T*DF8SF(Q,T3,14)
      DF7SF=0.7978845608028654D 00*(S2*DCOS(EP)+S1*DSIN(EP))/T2
      RETURN
      END
```

The UNIVERSITY OF ASTON
IN BIRMINGHAM
LIBRARY

FUNCTION RF8SF

```
      REAL FUNCTION RF8SF(A,T,N)
C  THIS FUNCTION EVALUATES CHEBYSHEV POLY
C  IN SINGLE PRECISION
C  A  COEFF OF CHEBYSHEV POLYNOMIAL
C  T  POINT AT WHICH POLYNOMIAL IS TO BE EVALUATED
C  N  DIMENSION OF ARRAY A
      REAL A,T,T2,B0,B1,B2
      DIMENSION A(N)
      T2=T+T
      B1=A(N)
      RF8SF=B1
      IF(N.LT.2) RETURN
      M=N-1
      B0=T2*B1+A(M)
      B2=0.0E 00
      IF(N.EQ.2) GO TO 20
      M2=M-1
      DO 10 I=1,M2
      J=M-I
      B2=B1
      B1=B0
10    B0=T2*B1-B2+A(J)
20    RF8SF=0.50E 00*(B0-B2)
      RETURN
      END
```

FUNCTION DF8SF

```
      DOUBLE PRECISION FUNCTION DF8SF(A,T,N)
C  THIS FUNCTION EVALUATES CHEBYSHEV POLYNOMIAL
C  IN DOUBLE PRECISION
C  A  COEFF OF POLYNOMIAL
C  T  POINT WHERE POLYNOMIAL IS TO BE EVALUATED
C  N  DIMENSION OF ARRAY A
      DOUBLE PRECISION A,T,T2,B0,B1,B2
      DIMENSION A(N)
      T2=T+T
      B1=A(N)
      DF8SF=B1
      IF(N.LT.2) RETURN
      M=N-1
      B0=T2*B1+A(M)
      B2=0.0D 00
      IF(N.EQ.2) GO TO 20
      M2=M-1
      DO 10 I=1,M2
      J=M-I
      B2=B1
      B1=B0
10    B0=T2*B1-B2+A(J)
20    DF8SF=0.50D 00*(B0-B2)
      RETURN
      END
```