P6060

BASIC Software Library

SSL (Scientific Subroutine Library)
Part II: Numerical Analysis
Programmer's Guide

Preliminary edition

olivetti

GP Code 3973570 G (0)

CONTENTS

PREFACE

This publication is addressed to P6060 users who develop their own application programs in the fields of mathematics, industry, science and engineering.

SUMMARY

This manual contains the reference documentation and the instructions for use of the subroutines of the P6060 Scientific Subroutine Library (SSL) - Part 2 - Numerical Analysis.
The subroutines have been classified according to the Standard Olivetti Software classification:

511 Generation of functions
512 Solution of equations
513 Curve fitting and interpolation.

RELATED DOCUMENTS:
P6060 Reference Manual code 3940920 Q

FIRST EDITION : January 1977

DISTRIBUTION : Licensed (L)

This material was prepared for the benefit of Olivetti Customers. It is recommended that the package be test run before actual use.

Anything in the standard form of the Olivetti Sales Contract to the contrary not withstanding all software being sold to Buyer is sold "as is". THERE ARE NO WARRANTIES EXPRESS OR IMPLIED INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTY OF MERCHANTABILITY AND/OR THE WARRANTY OF FITNESS FOR PURPOSE AND OLIVETTI SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES IN CONNECTION WITH SUCH SOFTWARE.

PUBLICATION ISSUED BY:

Olivetti
Direzione Marketing-Servizio Documentazione
77, Via Jervis - 10015 IVREA (Italy)

Purpose and objectives of SSL

The purpose of the P6060 Scientific Subroutine Library is to provide the user who develops his own application programs with a set of prewritten mathematical and statistical library routines which can easily and automatically be incorporated into his work to build a single program unit.
The routines are designed to solve the more common computational problems in mathematics and statistics.
The routines are free of input/output, allowing the programmer to use one routine, or more routines in the sequence that fit his computational needs, and to apply them to the particular input/output structure requested by the problem.

How to use the Scientific Subroutine Library

The common characteristics of the routines of the Olivetti P6060 Scientific Subroutine Library are:

- the routines do not include input/output statements, and operate on data already in memory
- the routines do not contain fixed maximum dimensioning for the data array on which they operate
- the routines are written in BASIC language and have the format of user defined functions
- generally the routines do not require the use of STRING or MAT options; a few cases in which the MAT option is required are clearly signalled
- the routines are stored in a user floppy disk, in the Package Library, each routine in a single text file having a filename *SL followed by a mnemonic related with the computational method
- the routines are documented uniformly; the use of each routine is illustrated by a simple program containing input of data, routine call and print out of results.

Before calling the routine, the user must accomodate the data on which the routine has to operate taking into account that data may be:

- single numerical variables: they can be used as function parameters, and the user may choose the names he likes, and write these names as parameters of the function in the calling statement
- numerical arrays (one or two dimensions): in this case they are global variables for the program and the routine and arrays must be dimensioned in the user program, declared single or double precision as the problem requires, and must have tha same names

used in the function. Names commonly used are E, F, G, H.

The results of the routines may be:

- only one simple numerical variable: in this case it will be given by the return value of the function
- more than one simple numerical variables or numerical arrays: in this case they will be global variables; the names commonly used are P1, P2 ... Q1, Q2 ... for simple numerical variables , and E, F, G, H for arrays. These arrays must be dimensioned and declared in the user program.

In order to build a BASIC program incorporating one or more routines of the P6060 SSL, the user should write the main program taking into account the following rules:

- the program must contain input and output statements using, for global variables, the same names used in the routines
- the program must contain the necessary DIM and DCL statements for the global arrays
- the SSL routine is referenced, in the calling statement, by a user defined name FN (letter) ....; the user may choose one of the 26 allowed names (FNA, FNB ... FNZ) to be associated to a specific SSL routine in the particular program
- when several SSL routines are incorporated in the same program, they must be associated to different function names
- the END statement (usually to be placed at line number 9999 : 9999 END) must not be present in the program.

The procedure to incorporate the SSL routines into the user program is:

- switch on the system
- load a system disk having some space to store the user program and the user disk containing the SSL library
- enter NEW, then enter the main program
- enter the LINK command for the first called SSL routine, with the format:

  LINK filename, letter

  when filename is the name of the file in which the routine is stored : *SL...
  letter is the letter used in the main program to reference the routine FN (letter)
- enter the LINK command for each of the other SSL routines called, if any, specifying for each routine the corresponding filename and letter. In the case of calling SSL routines stored in different user disks, after having linked the routines of the present disk enter the command DCHANGE U, then load another SSL floppy disk, press CONTINUE and go on entering the necessary LINK commands
- enter the END statement (usually : 9999 END)
- save the program on the system disk.

After the linking procedure is completed, the SSL floppy disk may be removed from the system and filed. The linked SSL routines are compiled together with the calling program leading to a program unit which will reside similtaneously in the memory, and may be run, listed, modified, saved as any other user program.
The documentation of each SSL routines provides both the information relevant to the used computational method, and the information necessary for a correct use of the routine itself .
The listing of the routine is included, for possible modifications.
The routines are referenced by the name of the files in which they are stored, that name beginning with the prefix *SL.
The following structure is used in the SSL documentation:

	Subroutine's name *SL....
Title	(Subroutine's title)
Purpose	(A brief description of the subroutine's purpose and features)
Method	(The used formula, or a brief description of the algorithm or method used)
Calling statement	F = FNZ (list of function parameters)
Parameters	(The parameters are listed with their meaning)
Global variables:	
- input	(The program variables on whose values the subroutine operates are listed, with their meaning, specifying if arrays, one or two dimensions, or simple variables)
- returned	(The program variables whose values are set or modified by the subroutine are listed, with their meaning and dimensions)
Status value	(List of the possible values returned by the function, with their meaning: in general Ø means OK, an integer > Ø means error)
Listing	(Listing of the subroutine)
Example of use	(A listing of a simple program showing the use of the subroutine, consisting of an input routine, one or more subroutines calls, and a print out routine; the listing is made before the subroutine's insertion by means of LINK command. Then the used LINK command is shown, and a sample run of the program, with typical input data, follows.)

Elementary Functions (real)

Elementary Functions (complex)

Polynomials

Higher mathematical functions

*SLATN2

Title	Arctangent of a ratio
Purpose	To compute the arctangent of a ratio of two real numbers y/x in the range $(-\pi, \pi)$
Calling Statement	F = FNZ (X, Y)
Parameters	X denominator Y numerator
Global variables	None
Function value	The arctangent of y/x

Listing

```
OLD *SLATN2
LIST
FILE    *SLATN2

0002 DEF FNA(X,Y)T
0004 IF X=0 THEN 20
0006 LET T=ATN(Y/X)
0008 IF X>0 THEN 22
0010 IF Y>=0 THEN 16
0012 LET T=T-PI
0014 GOTO 22
0016 LET T=T+PI
0018 GOTO 22
0020 LET T=SGN(Y)*PI/2
0022 LET FN*=T
0024 FNEND

END OF LISTING
```

Example of use

```
OLD RUATN2
LIST
FILE     RUATN2


0010 DISP "ENTER X AND Y";
0020 INPUT X,Y
0030 PRINT
0040 PRINT "X=";X,"Y=";Y
0050 PRINT "arctangent=";FNA(X,Y)
0060 GOTO 10

END OF LISTING


LINK *SLATN2,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

X= 0 Y= 0
arctangent= 0

X= 0 Y= 1
arctangent= 1.5707963

X= 1 Y= 0
arctangent= 0

X= 0 Y=-1
arctangent=-1.5707963

X= 1 Y= 1
arctangent= .78539816

X= 1 Y=-1
arctangent=-.78539816

X=-1 Y= 1
arctangent= 2.3561945

X= 1.0000000E+08 Y= 1.0000000E+08
arctangent= .78539816
```

*SLCONV

Title	Reduction of an angle to the first quadrant
Purpose	To reduce an angle (measured in radians) to the range $(-\pi, \pi)$
Calling Statement	F = FNZ (T)
Parameters	T = angle (radians)
Global variables	None
Function value	The resultant value of the angle

Listing

```
OLD *SLCONV
LIST
FILE    *SLCONV

0010 DEF FNA(T)P2
0020 LET P2=PI*2
0030 LET T=T-INT(T/P2)*P2
0040 IF T>PI THEN 70
0050 IF T<=-PI THEN 70
0060 GOTO 80
0070 LET T=T-T/T*P2
0080 LET FN*=T
0090 FNEND

END OF LISTING
```

Example of use

```
OLD RNCONV
LIST
FILE      PUCONV


0005 PRINT
0010 PRINT "ANGLE","TRANSFORMED ANGLE"
0020 DISP "ENTER ANGLE (radians)";
0025 INPUT T
0030 PRINT
0040 PRINT T,FNA(T)
0050 GOTO 20

END OF LISTING



LINK *SLCONV.A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

ANGLE             TRANSFORMED ANGLE

 1                 1

 2                 2

-3.1415927         3.1415926

-3.16              3.1231853

-6.3              -1.6814693E-02

 6.3               1.6814693E-02

 8962              2.1777520

-8962             -2.1777520

 6.2831853        -7.1800000E-09

 546              -.63712172
```

*SLRPCC

Title	Rectangular to polar coordinates conversion
Purpose	To convert a complex number from rectangular to polar coordinates
Method	Modulus $\varrho = \sqrt{x^2 + y^2}$
	phase $\vartheta = \operatorname{atn}\left(\frac{y}{x}\right)$
Calling Statement	F = FNZ (x,y)
Parameters	X real part of the complex number
	Y imaginary part
Global variables returned	P modulus
	Q phase $(-\pi < \varrho < \pi)$
Status value	ø

Listing

```
OLD *SLRPCC
LIST
FILE     *SLRPCC

0002 DEF FNR(X,Y)
0004 LET P=SQR(X*X+Y*Y)
0006 IF X=0 THEN 22
0008 LET Q=ATN(Y/X)
0010 IF X>0 THEN 24
0012 IF Y>=0 THEN 18
0014 LET Q=Q-PI
0016 GOTO 24
0018 LET Q=Q+PI
0020 GOTO 24
0022 LET Q=SGN(Y)*PI/2
0024 LET FN*=0
0026 FNEND

END OF LISTING
```

Example of use

```
OLD RURPCC
LIST
FILE     RURPCC

0010 DISP "ENTER X AND Y";
0020 PRINT
0030 INPUT X,Y
0040 PRINT "X=";X;"Y=";Y
0050 LET Z=FNA(X,Y)
0060 PRINT "RO=";P;"TETA=";Q
0070 GOTO 10

END OF LISTING


LINK *SLRPCC,A
9999 END

RUN
**** FORMALLY CORRECT PROGRAM ****

X=-1 Y=-1
RO= 1.4142136 TETA=-2.3561945

X= 1 Y= 0
RO= 1 TETA= 0

X= 0 Y= 1
RO= 1 TETA= 1.5707963

X= 0 Y=-1
RO= 1 TETA=-1.5707963

X= 0 Y=-100
RO= 100 TETA=-1.5707963

X=-1 Y= 0
RO= 1 TETA= 3.1415927
```

*SLPRCC

Title	Polar to rectangular coordinates conversion
Purpose	To convert a complex number from polar to rectangular coordinates
Method	$x = \varrho \cos \vartheta$ $y = \varrho \sin \vartheta$
Calling statement	F = FNZ (R, T)
Parameters	R = modulus of the complex number T = phase
Global variables returned	P1 real part P2 imaginary part
Status value	ø

Listing

```
FILE    *SLPRCC

0002 DEF FNZ(R,T)
0004 LET P1=R*COS(T)
0006 LET P2=R*SIN(T)
0008 LET FN*=0
0010 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RUPRCC

0010 DISP "ENTER RO AND TETA";
0020 PRINT
0030 INPUT R,T
0040 PRINT "RO=";R;"TETA=";T
0050 LET Z=FNA(R,T)
0060 PRINT "X=";P1;"Y=";P2
0070 GOTO 10

END OF LISTING


LINK *SLPRCC,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

RO= 1000 TETA= 236
X=-928.46012 Y=-371.43209

RO= 5 TETA= 0
X= 5 Y= 0

RO= 0 TETA= 0
X= 0 Y= 0

RO= 0 TETA= 25
X= 0 Y= 0
```

*SLCSIN

Title — Sine of a complex number

Purpose — To compute the sine of a complex number, expressed either in rectangular or in polar coordinates. The results are a complex number expressed in the same coordinates as the argument

Method — a) Rectangular coordinates

$$\text{Sin } Z = \sin(x + iy) =$$
$$= \sin(x)\cosh(y) + i\sinh(y)\cos(x)$$
$$= \sin(x)\left[\frac{e^{y} + e^{-y}}{2}\right] + i\cos(x)\left[\frac{e^{y} - e^{-y}}{2}\right]$$

b) polar coordinates

$$\varrho = \sqrt{\frac{1}{2}\left(\cosh(2y) - \cos(2x)\right)}$$

$$\vartheta = \arctan\left(\frac{\cos x}{\sin x}\,\frac{\sinh y}{\cosh y}\right)$$

Calling Statement — F = FNA (T, X, Y)

Parameters — T = type of coordinates: T = 0 rectangular; T = 1 polar

X, Y coordinates:

for T = 0: X = real part; Y = imaginary part

for T = 1: X = modulus; Y = phase

Global variables:
returned — P, Q results:

for T = 0: P = real part of the sine; Q = imaginary part of the sine

for T = 1 P = modulus of the sine

Q = phase of the sine

Status value ø

Other functions called : *SLANT2 used as FNZ

Listing

```
OLD *SLCSIN
LIST
FILE     *SLCSIN

0002 DEF FNA(T,X,Y)A,B
0004 IF T<>0 THEN 14
0006 LET A=EXP(Y)
0008 LET P=SIN(X)*(A+1/A)/2
0010 LET Q=COS(X)*(A-1/A)/2
0012 GOTO 22
0014 LET A=X*COS(Y)
0016 LET B=EXP(X*SIN(Y))
0018 LET P=SQR((B+B+1/(B*B))/4-COS(2*A)/2)
0020 LET Q=FNZ(SIN(A)*(B+1/B),COS(A)*(B-1/B))
0022 LET FN*=0
0024 FNEND

END OF LISTING
```

Example of use

```
OLD RUCSIN
LIST
FILE     RUCSIN


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa        =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : SIN(alfa) =  ##.##############      ##.##############

END OF LISTING


LINK *SLCSIN,A
LINK *SLATN2,Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0             alfa        = .52359878  0
 SIN(alfa) =   0.5000000038120       0.0000000000000

T= 0             alfa        = .52359878  1
 SIN(alfa) =   0.7715403232895       1.0177540856660

T= 0             alfa        = 0  1
 SIN(alfa) =   0.0000000000000       1.1752011936430

T= 1             alfa        = 0  0
 SIN(alfa) =   0.0000000000000       0.0000000000000

T= 1             alfa        = .52359878  0
 SIN(alfa) =   0.5000000038125       0.0000000000000

T= 1             alfa        = 1  0
 SIN(alfa) =   0.8414709848082       0.0000000000000
```

*SLCCOS

Title

Cosine of a complex number

Purpose

To compute the cosine of a complex number; both the argument and the results can be expressed either in rectangular or in polar coordinates

Method

a) Rectangular coordinates

$$\cos Z = \cos (x + iy) =$$

$$= \cos (x) \cosh (y) - i \sin (x) \sinh (y)$$

$$= \cos (x) \left[\frac{e^{y} + e^{-y}}{2}\right] + i \sin (x) \left[\frac{e^{-y} - e^{y}}{2}\right]$$

b) polar coordinates

$$R = \sqrt{\frac{1}{2}\left(\cosh (2y) - \cos(2x)\right)}$$

$$\vartheta = \arctan\left(\frac{\sin x}{\cos x}\,\frac{\sinh y}{\cosh y}\right)$$

Calling Statement

F = FNA (T, X, Y)

Parameters

T = type of coordinates: T = 0 rectangular
T = 1 polar

X, Y coordinates:

for T = 0 X = real part
Y = imaginary part

for T = 1 X = modulus
Y = phase

Global variables returned

P, Q results

for T = 0 P = real part of the cosine
Q = imaginary part of the cosine

for T = 1 P = modulus of the cosine
Q = phase of the cosine

Status value ø

Other functions called: *SLATN2 used as FNZ

Listing

```
OLD *SLCCOS
LIST
FILE    *SLCCOS


0002 DEF FNA(T,X,Y)A,B
0004 IF T<>0 THEN 14
0006 LET A=EXP(Y)
0008 LET P=COS(X)*(A+1/A)/2
0010 LET Q=SIN(X)*(1/A-A)/2
0012 GOTO 22
0014 LET A=X*COS(Y)
0016 LET B=EXP(X*SIN(Y))
0018 LET P=SQR((B*B+1/(B*B))/4+COS(2*A)/2)
0020 LET Q=FNZ(COS(A)*(B+1/B),SIN(A)*(1/B-B))
0022 LET FN*=0
0024 FNEND

END OF LISTING
```

Example of use

```
OLD RUCCOS
LIST
FILE     RUCCOS


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa       =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : COS(alfa) =  ##.##############     ##.##############

END OF LISTING


LINK *SLCCOS,A
LINK *SLATN2,Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0            alfa       = 0  0
 COS(alfa) =   1.0000000000000      0.0000000000000

T= 0            alfa       = .1  0
 COS(alfa) =   0.9950041652780      0.0000000000000

T= 1            alfa       = .1  0
 COS(alfa) =   0.9950041652780      0.0000000000000

T= 1            alfa       = .7  0
 COS(alfa) =   0.7648421872837      0.0000000000000
```

*SLCTAN

Title — Tangent of a complex number

Purpose — To compute the tangent of a complex number; both the argument and the results can be expressed in rectangular or in polar coordinates

Method — a) Rectangular coordinates:

$$\tan(z) = \tan(x + iy) =$$

$$= \frac{\sin(2x)}{\cos(2x) + \cosh(2y)} + i \frac{\sinh(2y)}{\cos(2x) + \cosh(2y)}$$

b) polar coordinates:

$$\varrho = \sqrt{\frac{\cosh(2y) - \cos(2x)}{\cosh(2y) + \cos(2x)}}$$

$$\vartheta = \arctan\left(\frac{\sinh(2y)}{\sin(2x)}\right)$$

Calling Statement — F = FNA (T, X, Y)

Parameters — T = type of coordinates: T = 0 rectangular; T = 1 polar

X, Y coordinates:

for T = 0: X = real part; Y = imaginary part

for T = 1: X = modulus; Y = phase

Global variables returned — P, Q results

for T = 0: P = real part of the tangent; Q = imaginary part of the tangent

for T = 1: P = modulus of the tangent; Q = phase of the tangent

Status value — Ø

Other functions called: *SLATN2 used as FNZ

Listing

```
LIST
FILE    *SLCTAN


0002 DEF FNA(T,X,Y)A,B
0004 IF T<>0 THEN 18
0006 LET X=2*X
0008 LET Y=EXP(2*Y)
0010 LET A=COS(X)+(Y+1/Y)/2
0012 LET P=SIN(X)/A
0014 LET Q=(Y-1/Y)/(2*A)
0016 GOTO 28
0018 LET A=2*X*COS(Y)
0020 LET B=EXP(2*X*SIN(Y))
0022 LET Y=1/B
0024 LET P=SGR(((B+Y)/2-COS(A))/((B+Y)/2+COS(A)))
0026 LET Q=FNZ(SIN(A),(B-Y)/2)
0028 LET FN*=0
0030 FNEND
END OF LISTING
```

Example for use

```
OLD RUCTAN
LIST
FILE     RUCTAN


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa       =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : TAN(alfa) =  ##.##############     ##.##############

END OF LISTING


LINK *SLCTAN,A
LINK *SLATN2,Z




9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0            alfa       = 1  0
 TAN(alfa) =   1.5574077246570      0.0000000000000

T= 0            alfa       = 1  1
 TAN(alfa) =   0.2717525853194      1.0839233273390

T= 1            alfa       = 1  0
 TAN(alfa) =   1.5574077246560      0.0000000000000

T= 1            alfa       = 2  1.57
 TAN(alfa) =   0.9640277146020      1.5706796058290
```

Title — Hyperbolic sine of a complex number

Purpose — To compute the hyperbolic sine of a complex number; both the argument and the results can be expressed either in rectangular or in polar coordinates

Method — Sinh (x + iy) = sinh x cos y + i cosh x sin y

$$\varrho = \sqrt{\frac{1}{2}\left(\cosh(2x) - \cos(2y)\right)}$$

$$\vartheta = \arctan\left(\frac{\cosh x \sin y}{\sinh x \cos y}\right)$$

Calling Statement — F = FNA (T,X,Y)

Parameters — T = type of coordinates: T = 0 rectangular; T = 1 polar

X, Y coordinates:

for T = 0: X = real part; Y = imaginary part

for T = 1: X = modulus; Y = phase

Global variables returned — P, Q results

for T = 0: P = real part of the hyperbolic sine; Q = imaginary part of the hyperbolic sine

for T = 1: P = modulus of the hyperbolic sine; Q = phase of the hyperbolic sine

Status value — Ø

Other functions called: *SLATN2 used as FNZ

Listing

```
OLD *SLCSIH
LIST
FILE    *SLCSIH


0010 DEF FNA(T,X,Y)A,B
0020 IF T<>0 THEN 70
0030 LET A=EXP(X)
0040 LET P=COS(Y)*(A-1/A)/2
0050 LET Q=SIN(Y)*(A+1/A)/2
0060 GOTO 110
0070 LET B=EXP(X*COS(Y))
0080 LET A=X*SIN(Y)
0090 LET P=SQR((B*B+1/(B*B))/4-COS(2*A)/2)
0100 LET Q=FNZ(COS(A)*(B-1/B);SIN(A)*(B+1/B))
0110 LET FN*=0
0120 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RUCSIH


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa       =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : SINH(alfa) =  ##.##############     ##.##############

END OF LISTING


LINK *SLCSIH,A
LINK *SLAIN2,Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0            alfa       = 0  0
 SINH(alfa) =   0.0000000000000      0.0000000000000

T= 0            alfa       = 1  0
 SINH(alfa) =   1.1752011936430      0.0000000000000

T= 0            alfa       = 0  1.5707963
 SINH(alfa) =   0.0000000000000      1.0000000000000

T= 1            alfa       = 1.4142  .78539816
 SINH(alfa) =   1.4453815282320      1.1158698562180
```

*SLCCOH

Title — Hyperbolic cosine of a complex number

Purpose — To compute the hyperbolic cosine of a complex number, both the arguments and the results can be expressed either in rectangular or in polar coordinates

Method — cosh (x + iy) = cosh X cos y + i sinh x sin y

$$\varrho = \sqrt{\frac{1}{2}(\cosh(2x)+\cos(2y))}$$

$$\vartheta = \operatorname{arctan}\left(\frac{\sinh x \sin y}{\cosh x \cos y}\right)$$

Calling Statement — F = FNA (T,X,Y)

Parameters — T = type of coordinates: T = 0 rectangular; T = 1 polar

X, Y coordinates:

for T = 0: X = real part; Y = imaginary part

for T = 1: X = modulus; Y = phase

Global variables returned — P, Q results

for T = 0: P = real part of the hyperbolic cosine; Q = imaginary part of the hyperbolic cosine

for T = 1: P = modulus of the hyperbolic cosine; Q = phase of the hyperbolic cosine

Status value — ∅

Other functions called: *SLATN2 used as FNZ

Listing

```
OLD *SLCCOH
LIST
FILE    *SLCCOH

0010 DEF FNA(T,X,Y)A,B
0020 IF T<>0 THEN 70
0030 LET A=EXP(X)
0040 LET P=COS(Y)*(A+1/A)/2
0050 LET Q=SIN(Y)*(A-1/A)/2
0060 GOTO 110
0070 LET B=EXP(X*COS(Y))
0080 LET A=X*SIN(Y)
0090 LET P=SQR((B*B+1/(B*B))/4+COS(2*A)/2)
0100 LET Q=FNZ(COS(A)*(B+1/B),SIN(A)*(B-1/B))
0110 LET FN*=0
0120 FNEND

END OF LISTING
```

Example of use

```
OLD RUCCOH
LIST
FILE     RUCCOH

0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa       =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : COSH(alfa) =  ##.##############     ##.##############

END OF LISTING.

LINK *SLCCOH,A
LINK *SLATN2,Z
9999 END
RUN RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0            alfa       = 0  0
 COSH(alfa) =   1.0000000000000      0.0000000000000

T= 0            alfa       = 1  0
 COSH(alfa) =   1.5430806348150      0.0000000000000

T= 0            alfa       = 0  1.5707963
 COSH(alfa) =  -0.0000000000063      0.0000000000000

T= 1            alfa       = 1.4142  .78539816
 COSH(alfa) =   1.2934443867120      0.8703144212289
```

*SLCTNH

Title — Hyperbolic tangent of a complex number

Purpose — To compute the hyperbolic tangent of a complex number, both the argument and the results can be expressed in rectangular or in polar coordinates

Method

$$\tanh(x+iy) = \frac{\sinh(2x)}{\cosh(2x)+\cos(2y)} + i\,\frac{\sin(2y)}{\cosh(2x)+\cos(2y)}$$

$$\varrho = \sqrt{\frac{\cosh(2x)-\cos(2y)}{\cosh(2x)+\cos(2y)}}$$

$$\vartheta = \arctan\left(\frac{\sin(2y)}{\sinh(2y)}\right)$$

Calling Statement — F = FNA (T, X, Y)

Parameters

T = type of coordinates: T = 0 rectangular; T = 1 polar

X, Y coordinates:

for T = 0: X = real part; Y = imaginary part

for T = 1: X = modulus; Y = phase

Global variables returned — P, Q results

for T = 0: P = real part of the hyperbolic tangent; Q = imaginary of the hyperbolic tangent

for T = 1: P = modulus of the hyperbolic tangent; Q = phase of the hyperbolic tangent

Status value — ø

Other functions called: *SLATN2 used as FNZ

Listing

```
OLD *SLCTNH
LIST
FILE     *SLCTNH


0010 DEF FNA(T,X,Y)A,B
0020 IF T<>0 THEN 90
0030 LET X=EXP(2*X)
0040 LET Y=2*Y
0050 LET A=(X+1/X)/2+COS(Y)
0060 LET P=(X-1/X)/(2*A)
0070 LET Q=SIN(Y)/A
0080 GOTO 140
0090 LET A=EXP(2*X*COS(Y))
0100 LET B=2*X*SIN(Y)
0110 LET Y=1/A
0120 LET P=SQR((A+Y)/2-COS(B))/((A+Y)/2+COS(B)))
0130 LET Q=FNZ((A-Y)/2,SIN(B))
0140 LET FN*=0
0150 FNEND

END OF LISTING
```

Example of use

```
OLD RUCTNH
LIST
FILE     RUCTNH


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa        =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : TANH(alfa) =  ##.##############     ##.##############

END OF LISTING


LINK *SLCTNH.A
LINK *SLATN2.Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0             alfa        = 0  0
 TANH(alfa) =    0.0000000000000      0.0000000000000

T= 0             alfa        = 1  0
 TANH(alfa) =    0.7615941559558      0.0000000000000

T= 0             alfa        = 0  1.5707963
 TANH(alfa) =  ****************     ****************

T= 0             alfa        = 1.4142  .78539816
 TANH(alfa) =    0.9930369681955      0.1178031332203
```

*SLCLN

Title Natural logarithm of a complex number

Purpose To compute the natural logarithm of a complex number; both the argument and the results can be expressed either in rectangular or in polar coordinates

Method

$$\log(z) = \log(x+iy) = \log\left(\sqrt{x^2+y^2}\right) + i \arctan\left(\frac{y}{x}\right)$$

$$\varrho_1 = |\log z| = |\log|z| e^{i\vartheta}| = \sqrt{(|\log|z||)^2+\vartheta^2}$$

$$\vartheta_1 = \arctan\left(\frac{\vartheta}{|z|}\right)$$

Calling Statement F = FNA (T, X, Y)

Parameters

T = type of coordinates: T = 0 rectangular
T = 1 polar

X, Y coordinates:

for T = 0 X = real part
Y = imaginary part

for T = 1 X = modulus
Y = phase

Global variables returned

P, Q results:

for T = 0 P = real part of logarithm
Q = imaginary part of logarithm

for T = 1 P = modulus of logarithm
Q = phase of logarithm

Status value ø

Other function called: *SLATN2 used as FNZ

Listing

```
OLD *SLCLN
LIST
FILE     *SLCLN


0002 DEF FNA(T,X,Y)
0004 IF T<>0 THEN 12
0006 LET P=LOG(SQR(X*X+Y*Y))
0008 LET Q=FNZ(X,Y)
0010 GOTO 18
0012 LET X=LOG(X)
0014 LET P=SQR(X*X+Y*Y)
0016 LET Q=FNZ(X,Y)
0018 LET FN*=Q
0020 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE      RUCLN


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 80
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa        =";X,Y
0120 LET Z=FNA(T,X,Y)
0130 PRINT USING 170,P,Q
0140 GOTO 10
0170 : LOG(alfa) =   ##.##############      ##.##############

END OF LISTING



LINK *SLCLN,A
LINK *SLATN2,Z
9999 END
RUN

++++ FORMALLY CORRECT PROGRAM ++++

T= 0             alfa        = 1   0
 LOG(alfa) =    0.0000000000000       0.0000000000000

T= 1             alfa        = 1   0
 LOG(alfa) =    0.0000000000000       0.0000000000000

T= 0             alfa        = 2   0
 LOG(alfa) =    0.6931471805592       0.0000000000000

T= 0             alfa        = 3   0
 LOG(alfa) =    1.0986122886670       0.0000000000000

T= 0             alfa        = 100   0
 LOG(alfa) =    4.6051701859880       0.0000000000000
```

*SLCEXP

Title	Exponential of a complex number
Purpose	To compute the exponential of a complex number; both the argument and the results can be expressed in rectangular or in polar coordinates
Method	let $z = x + iy = e^{z}$ $Re(e^{z}) = e^{x}\cos y$ $Im(e^{z}) = e^{x}\sin y$ $e^{z} = e^{x} = e^{\cos}$ $arg(e^{z}) = y = \sin$
Calling Statement	F = FNA (T, X, Y,)
Parameters	T = type of coordinates: T = 0 rectangular; T = 1 polar X, Y coordinates: for T = 0: X = real part; Y = imaginary part for T = 1: X = modulus; Y = phase
Global variables returned	P, Q results: for T = 0: P = real part of e^{z}; Q = imaginary part of e^{z} for T = 1: P = modulus of e^{z}; Q = phase of e^{z}
Status value	ø

Listing

```
LIST
FILE    *SLCEXP

0002 DEF FNA(T,X,Y)
0004 IF T<>0 THEN 14
0006 LET X=EXP(X)
0008 LET P=X*COS(Y)
0010 LET Q=X*SIN(Y)
0012 GOTO 18
0014 LET P=EXP(X*COS(Y))
0016 LET Q=X*SIN(Y)
0018 LET FN*=0
0020 FNEND

END OF LISTING
```

Example of use

```
OLD RUCEXP
LIST
FILE     RUCEXP

0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T
0110 PRINT "alfa       =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : EXP(alfa) =  ##.##############     ##.##############

END OF LISTING
```

```
LINK *SLCEXP.A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0              alfa        = 1  0
 EXP(alfa) =    2.7182818284590       0.0000000000000

T= 0              alfa        = 0  3.1415927
 EXP(alfa) =   -1.0000000000000      -0.0000000464076

T= 0              alfa        = 2  .52359878
 EXP(alfa) =    6.3991102754030       3.6945280776340

T= 1              alfa        = 1  0
 EXP(alfa) =    2.7182818284590       0.0000000000000

T= 1              alfa        = 2  1.57
 EXP(alfa) =    1.0015939223700       1.9999993658630
```

Title — Reciprocal of a complex number

Purpose — To compute the reciprocal of a complex number; both the argument and the results can be expressed either in rectangular or in polar coordinates

Method

$$\frac{1}{z} = \frac{1}{x+iy} = \frac{x-iy}{x^2+y^2} = \frac{x}{x^2+y^2} - i\,\frac{y}{x^2+y^2}$$

$$\varrho_1 = \left|\frac{1}{\varrho e^{i\vartheta}}\right| = \frac{1}{\varrho}$$

$$\vartheta_1 = -\vartheta$$

Calling Statement — F = FNA (T,X,Y)

Parameters

T = type of coordinates: T = 0 rectangular; T = 1 polar

X, Y coordinates:

for T = 0: X = real part; Y = imaginary part

for T = 1: X = modulus; Y = phase

Global variables returned

P, Q results:

for T = 0: P = real part of the reciprocal; Q = imaginary of the reciprocal

for T = 1: P = modulus of the reciprocal; Q = phase of the reciprocal

Status value — Ø

Listing

```
OLD *SLCRZ
LIST
FILE    *SLCRZ


0002 DEF FNA(T,X,Y)R
0004 IF T<>0 THEN 14
0006 LET R=X*X+Y*Y
0008 LET P=X/R
0010 LET Q=-Y/R
0012 GOTO 18
0014 LET P=1/X
0016 LET Q=-Y
0018 LET FN*=0
0020 FNEND

END OF LISTING
```

Example of use

```
OLD RUCRZ
LIST
FILE     RUCRZ


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T,
0110 PRINT "alfa       =";X;Y
0130 LET Z=FNA(T,X,Y)
0150 PRINT USING 170,P,Q
0160 GOTO 10
0170 : 1/(alfa) =  ##.##############     ##.##############

END OF LISTING


LINK *SLCRZ,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0            alfa       = 0  10
 1/(alfa) =   0.0000000000000     -0.1000000000000

T= 0            alfa       = 10  0
 1/(alfa) =   0.1000000000000      0.0000000000000

T= 0            alfa       = 3.75  2.85
 1/(alfa) =   0.1647265539204     -0.1295848890841
```

*SLCZMZ

Title Multiplication of two complex numbers

Purpose To compute the product of two complex numbers, both the argument and the results can be expressed either in rectangular or in polar coordinates

Method

$$Z_1 Z_2 = (X_1 + iY_1)(X_2 + iy_2) = X_1 X_2 - Y_1 Y_2 + i (Y_1 X_2 + Y_2 X_1)$$

$$\varrho = |Z_1 Z_2| = \varrho_1 \varrho_2$$

$$\vartheta = \vartheta_1 + \vartheta_2$$

Calling Statement F = FNA (T, X1, Y1, X2, Y2)

Parameters

T = type of coordinates: T = 0 rectangular
T = 1 polar

X1,Y1,X2,Y2 coordinates:

for T = 0
- X1 = real part of first number
- Y1 = imaginary part of first number
- X2 = real part of second number
- Y2 = imaginary part of second number

for T = 1
- X1 = modulus of first number
- Y1 = phase of first number
- X2 = modulus of second number
- Y2 = phase of second number

Global variables returned P, Q results

for T = 0
- P = real part of the sine
- Q = imaginary part of the sine

for T = 1
- P = modulus of the sine
- Q = phase of the sine

Status value Ø

Other function called: *SLCONV used as FNZ

Listing

```
LIST
FILE    *SLCZMZ

0002 DEF FNA(T,X1,Y1,X2,Y2)
0004 IF T<>0 THEN 12
0006 LET P=X1*X2-Y1*Y2
0008 LET Q=X1*Y2+Y1*X2
0010 GOTO 18
0012 LET P=X1*X2
0014 LET Q=Y1+Y2
0016 LET Q=FNZ(Q)
0018 LET FN*=0
0020 FNEND

END OF LISTING
```

Example of use

```
FILE    RUCZMZ

0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR";
0020 INPUT T
0030 IF T=1 THEN 90
0040 DISP "ENTER X AND Y OF THE 1ST No";
0050 INPUT X1,Y1
0060 DISP "ENTER X AND Y OF THE SEC.No";
0070 INPUT X2,Y2
0080 GOTO 140
0090 DISP "ENTER RO AND TETA OF THE 1ST No";
0100 INPUT X1,Y1
0110 DISP "ENTER RO AND TETA OF THE SEC.No";
0120 INPUT X2,Y2
0140 PRINT
0150 PRINT "T=";T
0160 PRINT "alfa     =";X1,Y1
0170 PRINT "beta     =";X2,Y2
0180 LET Z=FNA(T,X1,Y1,X2,Y2)
0190 PRINT "alfa*beta=";P,Q
0200 GOTO 10

END OF LISTING
```

```
OLD.RUCZMZ
LINK *SLCZMZ,A
LINK *SLCONV,2
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0
alfa      = 2      2
beta      = 1     -4
alfa*beta= 10     -6

T= 1
alfa      = 3      6
beta      = 2      3
alfa*beta= 6      2.7168147

T= 0
alfa      = 14     25
beta      = 52     32
alfa*beta=-72     1748

T= 0
alfa      = 0      0
beta      = 12     35
alfa*beta= 0      0

T= 1
alfa      = 3      0
beta      = 1      0
alfa*beta= 3      0
```

*SLCZDZ

Title — Division of two complex numbers

Purpose — To compute the ratio of two complex numbers; both the arguments and the results can be expressed either in rectangular or in polar coordinates

Method

let $z_1 = a + ib = e$; $z_2 = c + id = e$

$$\frac{z_1}{z_2} = \frac{a+ib}{e+id} \cdot \frac{c-id}{c-id} = \frac{(ac+bd)+i(bc-ad)}{c^2+d^2}$$

$$\frac{z_1}{z_2} =$$

$$\arg\left(\frac{z_1}{z_2}\right) =$$

Calling statement — F = FNA (T, X1, Y1, X2, Y2)

Parameters

T = type of coordinates: T = 0 rectangular; T = 1 polar

X1, Y1, X2, Y2 = coordinates:

for T = 0
- X1 = real part of z_1
- Y1 = imaginary part of z_1
- X2 = real part of z_2
- Y2 = imaginary part of z_2

For T = 1
- X1 = modulus of z_1
- Y1 = phase of z_1
- X2 = modulus of z_2
- Y2 = phase of z_2

Global variables returned — P, Q results

for T = 0
- P = real part of ratio
- Q = imaginary part of ratio

for T = 1
- P = modulus of ratio
- Q = phase of ratio

Status value — ø

Other function called: *SLCONV used as FNZ

Listing

```
OLD *SLCZDZ
LIST
FILE    *SLCZDZ

0002 DEF FNA(T,X1,Y1,X2,Y2)A,B,C
0004 IF T<>0 THEN 18
0006 LET A=X1*X2+Y1*Y2
0008 LET B=Y1*X2-X1*Y2
0010 LET C=X2*X2+Y2*Y2
0012 LET P=A/C
0014 LET Q=B/C
0016 GOTO 24
0018 LET P=X1/X2
0020 LET Q=Y1-Y2
0022 LET Q=FNZ(0)
0024 LET FN*=0
0026 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RRCZDZ

0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR";
0020 INPUT T
0030 IF T=1 THEN 90
0040 DISP "ENTER X AND Y OF THE 1ST No";
0050 INPUT X1,Y1
0060 DISP "ENTER X AND Y OF THE SEC.No";
0070 INPUT X2,Y2
0080 GOTO 140
0090 DISP "ENTER RO AND TETA OF THE 1ST No";
0100 INPUT X1,Y1
0110 DISP "ENTER RO AND TETA OF THE SEC.No";
0120 INPUT X2,Y2
0140 PRINT
0150 PRINT "T=";T
0160 PRINT "alfa      =";X1,Y1
0170 PRINT "beta      =";X2,Y2
0180 LET Z=FNA(T,X1,Y1,X2,Y2)
0190 PRINT "alfa/beta=";P,Q
0200 GOTO 10

END OF LISTING
```

```
LINK *SLCZDZ.A
LINK *SLCONV,Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0
alfa      = 2      2
beta      = 1     -4
alfa/beta=-.35294118              .58823529

T= 0
alfa      = 4      0
beta      = 2      0
alfa/beta= 2      0

T= 1
alfa      = 10     3.14
beta      = 2      3.14
alfa/beta= 5      0

T= 1
alfa      = 2      3.15
beta      = 12     3.15
alfa/beta= .16666667              0
```

*SLCSQR

Title Square root of a complex number

Purpose To compute the square root of a complex number; both the argument and the results can be expressed either in rectangular or in polar coordinates

Method

let $z = x + iy = \ell^{i}$

$\mathrm{Re}(\sqrt{z}) = \sqrt{\frac{+x}{\ell}}$

$\mathrm{Im}(\sqrt{z}) = \frac{y}{y}\sqrt{\frac{-x}{\ell}}$

$\sqrt{z} = \sqrt{}$

$\arg(\sqrt{z}) = \overline{\ell}$

Calling statement F = FNA (T, X, Y)

Parameters

T = type of coordinates:
- T = 0 rectangular
- T = 1 polar

X, Y = coordinates:

for T = 0
- X = real part
- Y = imaginary part

for T = 1
- X = modulus
- Y = phase

Global variables returned:

P, Q results:

for T = 0
- P = real part of $\sqrt{z}$
- Q = imaginary part of $\sqrt{z}$

for T = 1
- P = modulus of $\sqrt{z}$
- Q = phase of $\sqrt{z}$

Status value ∅

Other function called: *SLCONV used as FNZ

Listing

```
LIST
FILE     *SLCSQR

0002 DEF FNB(T,X,Y)R
0004 IF T<>0 THEN 14
0006 LET R=SQR(X*X+Y*Y)
0008 LET P=SQR((R+X)/2)
0010 LET Q=SGN(Y)*SQR((R-X)/2)
0012 GOTO 20
0014 LET P=SQR(X)
0016 LET Q=Y/2
0018 LET Q=FNZ(Q)
0020 LET FNB=Q
0022 FNEND

END OF LISTING
```

Example of use

```
OLD RUCSQR
LIST
FILE     RUCSQR

0010 DISP "ENTER T T=0 RECT.COORD.  T=1 POLAR COORD.";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y                          ";
0050 INPUT X,Y
0060 GOTO 100
0070 DISP "ENTER RO AND TETA                      ";
0080 INPUT X,Y
0100 PRINT
0110 PRINT "T=";T
0120 PRINT "alfa          =";X,Y
0130 LET Z=FNA(T,X,Y)
0140 PRINT "SQR(alfa)     =";P,Q
0150 LET A=P
0160 LET B=Q
0170 LET Z=FNB(T,A,B,A,B)
0180 PRINT "(SQR(alfa))↑2=";P,Q
0190 GOTO 10

END OF LISTING
```

```
LINK *SLCSQR.A
LINK *SLCZMZ.B
LINK *SLCGNU.Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0
alfa           = 1                  1
SQR(alfa)      = 1.0986841          .45598886
(SQR(alfa))↑2= 1.0000000            1.0000000

T= 0
alfa           =-13                 57
SQR(alfa)      = 4.7677911          5.9776109
(SQR(alfa))↑2=-13                   57.000000

T= 1
alfa           = 1                 -56.235633
SQR(alfa)      = 1                 -2.9850753
(SQR(alfa))↑2= 1                    .31303476

T= 1
alfa           = 9                  0
SQR(alfa)      = 3                  0
(SQR(alfa))↑2= 9                    0

T= 1
alfa           = 9                  3.141592
SQR(alfa)      = 3                  1.570796
(SQR(alfa))↑2= 9                    3.141592
```

*SLCZN

Title	Integral power of a complex number (Z^N recurrence)
Purpose	To compute the power Z^N with Z complex and N integer ≥ 0
Method	let $Z = x + iy = e^i$ $Z^m = Z \cdot Z^{m-1}$, the following recurrence is used: $Z^m = X_m + i\, y_m$ where $X_m = X \cdot X_{m-1} - y \cdot y_{m-1}$ $\quad X_0 = 1$ $y_m = X\, y_{m-1} + y \cdot X_{m-1}$ $\quad y_0 = 0$ $\lvert Z^m \rvert = {}^m$ $\arg (Z^m) = m$
Calling statement	F = FNA (T, N, X, Y)
Paramters	T = type of coordinates: $\quad$ T = 0 rectangular T = 1 polar N = exponent X, Y coordinates: for T = 0 $\quad$ X = real part Y = imaginary part for T = 1 $\quad$ X = modulus Y = phase
Global variables returned	P, Q results for T = 0 $\quad$ P = real part of Z^N Q = imaginary part of Z^N for T = 1 $\quad$ P = modulus of Z^N Q = phase of Z^N
Status value	ø

Other function called: *SLCONV used as FNZ

Listing

```
LIST
FILE     *SLCZN


0002 DEF FNA(T,N,X,Y)I,E1,F1
0004 LET P=1
0006 LET Q=0
0008 IF N=0 THEN 38
0010 IF T<>0 THEN 30
0012 FOR I=1 TO N STEP 1
0014 LET E1=P
0016 LET F1=Q
0018 LET P=X*E1-Y*F1
0020 LET Q=X*F1+Y*E1
0022 NEXT I
0026 GOTO 38
0030 FOR I=1 TO N STEP 1
0032 LET P=P*X
0034 NEXT I
0036 LET Q=FNZ(Y*N)
0038 LET FN*=0
0040 FNEND

END OF LISTING
```

Example of use

```
OLD RUCZN
LIST
FILE      RUCZN


0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 100
0070 DISP "ENTER R0 AND TETA";
0080 INPUT X,Y
0100 PRINT
0110 PRINT "T=";T
0120 PRINT "alfa       =";X,Y
0130 DISP "ENTER EXPONENT N";
0140 INPUT N
0150 FOR I=0 TO N STEP 1
0160 LET Z=FNA(T,I,X,Y)
0190 PRINT "alfa ↑";I;"  =";P,Q
0200 NEXT I
0230 GOTO 10

END OF LISTING
```

```
LINK *SLCZN,A
LINK *SLCONU,Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0
alfa          = 2    0
alfa ↑ 0      = 1    0
alfa ↑ 1      = 2    0
alfa ↑ 2      = 4    0
alfa ↑ 3      = 8    0
alfa ↑ 4      = 16                    0
alfa ↑ 5      = 32                    0
alfa ↑ 6      = 64                    0
alfa ↑ 7      = 128                   0
alfa ↑ 8      = 256                   0
alfa ↑ 9      = 512                   0

T= 1
alfa          = 1    1
alfa ↑ 0      = 1    0
alfa ↑ 1      = 1    1
alfa ↑ 2      = 1    2
alfa ↑ 3      = 1    3
alfa ↑ 4      = 1   -2.2831853
alfa ↑ 5      = 1   -1.2831853
alfa ↑ 6      = 1   -.28318531
alfa ↑ 7      = 1    .71681469

T= 0
alfa          = 1    1
alfa ↑ 0      = 1    0
alfa ↑ 1      = 1    1
alfa ↑ 2      = 0    2
alfa ↑ 3      =-2    2
alfa ↑ 4      =-4    0
alfa ↑ 5      =-4   -4
alfa ↑ 6      = 0   -8
alfa ↑ 7      = 8   -8
```

*SLCZA

Title Real power of a complex number

Purpose To compute Z^a, with Z complex and a real number; both the argument and the results can be expressed either in rectangular or in polar coordinates

Method

let $z = x + iy = e^{i}$

$Re(z^a) = {}^{a}\cos(a)$

$Im(z^a) = {}^{a}\sin(a)$

$z^a = {}^{a}$

$arg(z^a) = a$

Calling statement F = FNA (T, A, X, Y)

Parameters

T = type of coordinates: T = 0 rectangular
T = 1 polar

A = exponent

X,Y= coordinates:

for T = 0
X = real part
Y = imaginary part

for T = 1
X = modulus
Y = phase

Global variables returned

P, Q results:

for T = 0
P = real part of Z^a
Q = imaginary part of Z^a

for T = 1
P = modulus of Z^a
Q = phase of Z^a

Status value ø

Other function called *SLATN2 used as FNZ

Listing

```
FILE    *SLCZA

0002 DEF FNA(T,A,X,Y)R
0004 IF T=1 THEN 16
0006 LET R=SQR(X*X+Y*Y)↑A
0008 LET Q=FNZ(X,Y)*A
0010 LET P=R*COS(Q)
0012 LET Q=R*SIN(Q)
0014 GOTO 20
0016 LET P=X↑A
0018 LET Q=Y*A
0020 LET FN*=Q
0022 FNEND

END OF LISTING
```

Example of use

```
FILE    RUCZA

0010 DISP "ENTER 0=RECTANGULAR OR 1=POLAR ";
0020 INPUT T
0030 IF T=1 THEN 70
0040 DISP "ENTER X AND Y";
0050 INPUT X,Y
0060 GOTO 90
0070 DISP "ENTER RO AND TETA";
0080 INPUT X,Y
0090 PRINT
0100 PRINT "T=";T
0110 PRINT "alfa       =";X,Y
0120 DISP "ENTER EXPONENT n";
0130 INPUT N
0140 LET Z=FNA(T,N,X,Y)
0150 PRINT USING 170,N,P,Q
0160 GOTO 10
0170 :alfa↑##.#########↑↑↑↑ =  ##.##############   ##.##############

END OF LISTING
```

```
LINK *SLCZA.A
LINK *SLATN2.Z
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

T= 0
alfa        = 2   0
alfat 3.000000E+00 =   8.0000000000000    0.0000000000000

T= 1
alfa        = 2   0
alfat 3.000000E+00 =   8.0000000000000    0.0000000000000

T= 0
alfa        = 52                  0
alfat 6.000000E+00 =  ***************    0.0000000000000

T= 1
alfa        = 1   1
alfat 2.700000E+00 =   1.0000000000000    2.7000000000000

T= 1
alfa        = 2   2
alfat 3.140000E+00 =   8.8152409269990    6.2800000000000
```

*SLPLCC

Title	Evaluation of complex polynomials (compl. arguments)
Purpose	To evaluate a polynomial with complex coefficients and complex arguments
Method	Horner Scheme
Calling Statement	F = FNA (X, Y)
Parameters	X real part Y imaginary part of the argument

Global variable:

- input	U (I)	array of real part of the coefficients
	V (I)	array of imaginary part of the coefficients
	D	degree
- return	P	real part
	Q	imaginary part of the value

Status value ø

Listing

```
LIST
FILE    *SLPLCC

0010 DEF FNA(X,Y)I,Q1,P1
0020 LET P1=Q1=0
0030 FOR I=D+1 TO 1 STEP -1
0040 LET P=P1*X-Q1*Y+U(I)
0050 LET Q=Q1*X+P1*Y+V(I)
0060 LET P1=P
0070 LET Q1=Q
0080 NEXT I
0090 LET FN*=0
0100 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE    +PLCC

0010 DCL 30(B$,C$)
0020 PRINT
0030 DIM U(100),V(100)
0040 PRINT "EVALUATION OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS";
0050 PRINT " AND COMPLEX ARGUMENTS"
0060 PRINT
0070 DISP "DEGREE OF THE POLYNOMIAL";
0080 INPUT D
0090PRINT"THE COEFFICIENT C=U+iV IS ENTERED AS U,V AND THE ARGUMENT X+iY AS X,Y"
0100 PRINT
0110 PRINT "COEFFICIENTS:"
0120 PRINT
0130 FOR I=D+1 TO 1 STEP -1
0140 DISP "U(";I-1;"),V(";I-1;")";
0150 INPUT U(I),V(I)
0160 LET S=SGN(V(I))
0170 LET A(1)=43+S*S-S
0180 CONVERT A TO A$ LENGTH 1
0190 BUILD B$,U(I),A$,"i",ABS(V(I))
0200 PRINT "C(";I-1;")=";B$,
0210 NEXT I
0220 PRINT
0230 PRINT
0240 DISP " ENTER X+iY";
0250 INPUT X,Y
0260 LET F=FNA(X,Y)
0270 PRINT
0280 LET S=SGN(Y)
0290 LET A(1)=43+S*S-S
0300 CONVERT A TO A$ LENGTH 1
0310 BUILD B$,X,A$,"i",ABS(Y)
0320 LET S=SGN(0)
0330 LET A(1)=43+S*S-S
0340 CONVERT A TO A$ LENGTH 1
0350 BUILD C$,P,A$,"i",ABS(0)
0360 PRINT "RESULT : F(";B$;")=";C$

END OF LISTING


LINK *SLPLCC,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

EVALUATION OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS AND COMPLEX ARGUMENTS

THE COEFFICIENT C=U+iV IS ENTERED AS U,V AND THE ARGUMENT X+iY AS X,Y

COEFFICIENTS:

C( 6 )= 1 +i 1  C( 5 )= 2 +i 2  C( 4 )= 3 +i 3  C( 3 )= 4 +i 4  C( 2 )= 5 +i 5
C( 1 )= 6 +i 6  C( 0 )= 0 +i 0


RESULT : F( 1 +i 1 )=-30--i 14
```

*SLPLRC

Title	Evaluation of real polynomials (compl. arguments)
Purpose	To evaluate a polynomial with real coefficients and complex arguments
Method	Horner Scheme
Calling Statement	F = FNA (X, Y)
Parameters	X = real part of the argument Y = imaginary part of the argument
Global variables:	
- input	C ( ) vector of coefficients (in descending order) D degree
- return	P real part Q imaginary part of the value
Status value	ø

Listing

```
OLD *SLPLRC

LIST
FILE    *SLPLRC

0010 DEF FNA(X,Y)I,P1,Q1
0020 LET P1=Q1=0
0030 FOR I=D+1 TO 1 STEP -1
0040 LET P=P1*X-Q1*Y+C(I)
0050 LET Q=P1*Y+Q1*X
0060 LET P1=P
0070 LET Q1=Q
0080 NEXT I
0090 LET FN*=0
0100 FNEND

END OF LISTING
```

Example of use

```
OLD +PLRC
LIST
FILE    +PLRC


0010 PRINT
0020 DIM C(100)
0030 PRINT "EVALUATION OF A REAL POLYNOMIAL WITH COMPLEX ARGUMENT"
0040 PRINT
0050 DISP "ENTER DEGREE OF THE POLYNOMIAL";
0060 INPUT D
0070 PRINT "COEFFICIENTS:"
0080 PRINT
0090 FOR I=D+1 TO 1 STEP -1
0100 DISP "ENTER C(";I-1;")";
0110 INPUT C(I)
0120 PRINT "C(";I-1;")=";C(I),
0130 NEXT I
0140 DISP "THE ARG. X+iY IS ENTERED AS X,Y";
0150 INPUT X,Y
0160 LET F=FNA(X,Y)
0170 PRINT
0180 LET S=SGN(Q)
0190 LET A(1)=43+S*S-S
0200 CONVERT A TO A$ LENGTH 1
0210 BUILD C$,P,A$,"i",ABS(Q)
0220 LET S=SGN(Y)
0230 LET A(1)=43+S*S-S
0240 CONVERT A TO A$ LENGTH 1
0250 BUILD B$,X,A$,"i",ABS(Y)
0260 PRINT "RESULT : F(";B$;") =";C$

END OF LISTING


LINK *SLPLRC,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

EVALUATION OF A REAL POLYNOMIAL WITH COMPLEX ARGUMENT

COEFFICIENTS:

C( 4 )= 1       C( 3 )= 2       C( 2 )= 3       C( 1 )= 4       C( 0 )= 5

RESULT : F( 1 +i 1 ) = 1 +i 14
```

*SLPLRR

Title	Evaluation of real polynomials (real argument)
Purpose	Calculate the value of a polynomial with real coefficients and real arguments
Method	Horner Scheme
Calling Statement	PRINT FNA (X)
Parameter	X Argument

Global variables:

- input	D Degree
	C ( ) Vector of coefficients (in descending order)
- return	FNA (X)

Listing

```
LIST
FILE    *SLPLRR

0010 DEF FNA(X)I,Q
0020 LET Q=C(D+1)
0030 FOR I=D TO 1 STEP -1
0040 LET Q=Q*X+C(I)
0050 NEXT I
0060 LET FN*=Q
0070 FNEND

END OF LISTING
```

Example of use

```
OLD +PLRR
LIST
FILE    +PLRR


0010 PRINT
0020-PRINT TAB(15);"EVALUATION OF A POLYNOM WITH "
0030 PRINT TAB(11);"REAL COEFFICIENTS AND REAL ARGUMENTS"
0040 PRINT
0050 DIM C(100)
0060 DISP "ENTER DEGREE";
0070 INPUT D
0080 PRINT "COEFFICIENTS :"
0090 PRINT
0100 FOR I=D+1 TO 1 STEP -1
0110 DISP "ENTER COEFFICIENT C(";I-1;")";
0120 INPUT C(I)
0130 PRINT "C(";I-1;")=";C(I),
0140 NEXT I
0150 PRINT
0160 PRINT
0170 DISP "ENTER ARGUMENT X";
0180 INPUT X
0190 PRINT "RESULT : P(";X;")=";FNA(X)
0200 PRINT

END OF LISTING


LINK *SLPLRR,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

            EVALUATION OF A POLYNOM WITH
         REAL COEFFICIENTS AND REAL ARGUMENTS

COEFFICIENTS :

C( 2 )= 1       C( 1 )= 0       C( 0 )= 1

RESULT : P( 2 )= 5


            EVALUATION OF A POLYNOM WITH
         REAL COEFFICIENTS AND REAL ARGUMENTS

COEFFICIENTS :

C( 15 )= 1      C( 14 )= 2      C( 13 )= 3      C( 12 )= 4      C( 11 )= 5
C( 10 )= 6      C( 9 )= 7       C( 8 )= 8       C( 7 )= 9       C( 6 )= 10
C( 5 )= 11      C( 4 )= 12      C( 3 )= 13


**** FORMALLY CORRECT PROGRAM ****

            EVALUATION OF A POLYNOM WITH
         REAL COEFFICIENTS AND REAL ARGUMENTS

COEFFICIENTS :

C( 15 )= 1      C( 14 )= 2      C( 13 )= 3      C( 12 )= 4      C( 11 )= 5
C( 10 )= 6      C( 9 )= 7       C( 8 )= 8       C( 7 )= 9       C( 6 )= 10
C( 5 )= 11      C( 4 )= 22      C( 3 )= 33      C( 2 )= 44      C( 1 )= 55
C( 0 )= 66

RESULT : P( 5.5 )= 1.9043235E+11
```

*SLPRRR

Title	Calculating the coefficient of a polynomial from the roots
Purpose	To compute the coefficients of a real polynomial from roots
Method	The coefficients are computed by multiplying the roots and summing up the terms of the same degree

Calling statement = FNA (R)

Parameter R Number of roots

Global variables:

- input E ( ) Vector of roots
- return A ( ) Vector of coefficients (in descending order)

Status value ø

Listing

```
FILE    *SLPRRR

0010 DEF FNA(R)K,J
0020 FOR K=3 TO R+1 STEP 1
0030 LET A(K)=0
0040 NEXT K
0050 LET A(1)=-E(1)
0060 LET A(2)=1
0070 FOR J=2 TO R STEP 1
0080 FOR K=J+1 TO 2 STEP -1
0090 LET A(K)=A(K-1)-A(K)*E(J)
0100 NEXT K
0110 LET A(1)=-A(1)*E(J)
0120 NEXT J
0130 LET FN*=0
0140 FNEND

END OF LISTING
```

Example of use

```
FILE    +PRRR

0010 DIM A(100),E(99)
0020 PRINT "COMPUTATION OF THE COEFFICIENTS OF A REAL POLYNOM FROM ROOTS"
0030 PRINT
0040 PRINT
0050 DISP "ENTER # ROOTS";
0060 INPUT R1
0070 FOR I=1 TO R1 STEP 1
0080 DISP "ENTER E";I;
0090 INPUT E(I)
0100 NEXT I
0110 PRINT "ROOTS:"
0120 PRINT
0130 FOR I=1 TO R1 STEP 1
0140 PRINT E(I),
0150 NEXT I
0160 PRINT
0170 PRINT
0180 LET I5=FNA(R1)
0190 PRINT "COEFFICIENTS IN DECREASING ORDER: "
0200 PRINT
0210 FOR I=R1+1 TO 1 STEP -1
0220 PRINT A(I);
0230 NEXT I
0240 PRINT
0250 PRINT
0260 GOTO 30

END OF LISTING



LINK *SLPRRR,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
COMPUTATION OF THE COEFFICIENTS OF A REAL POLYNOM FROM ROOTS


ROOTS:

 1              2              3              4              5
 6

COEFFICIENTS IN DECREASING ORDER:

 1 -21  175 -735  1624 -1764  720
```

*SLCOCC

Title	Calculating of the coefficients of a complex polynomial from roots
Purpose	To compute the coefficients of a complex polynomial from roots
Method	The coefficients are computed by multiplying the roots and summing up the terms
Calling Statement	I = FNA (R)
Parameter	R Degree of the polynomial (= number of roots)

Global variables

input	E ( )	real part of roots
	F ( )	imaginary part of roots
output	A ( )	real part of coefficients
	B ( )	imaginary part of coefficients

Status value FN* = -1 degree ≤ ∅

Listing

```
LIST
FILE    *SLCOCC

0002 DEF FNA(R)K,J,U,V
0004 IF R>0 THEN 10
0006 LET FN*=-1
0008 GOTO 50
0010 FOR K=3 TO R+1 STEP 1
0012 LET A(K)=0
0014 LET B(K)=B(2)=0
0016 NEXT K
0018 LET A(1)=-E(1)
0020 LET B(1)=-F(1)
0022 LET A(2)=1
0024 FOR J=2 TO R STEP 1
0026 FOR K=J+1 TO 2 STEP -1
0028 LET U=A(K)
0030 LET V=B(K)
0032 LET A(K)=A(K-1)-U*E(J)+V*F(J)
0034 LET B(K)=B(K-1)-V*E(J)-U*F(J)
0036 NEXT K
0038 LET U=A(1)
0040 LET V=B(1)
0042 LET A(1)=-U*E(J)+V*F(J)
0044 LET B(1)=-U*F(J)-V*E(J)
0046 NEXT J
0048 LET FN*=0
0050 FNEND

END OF LISTING
```

Example of use

```
OLD +COCC
LIST
FILE    +COCC


0010 DISP "ENTER DEGREE";
0020 INPUT R
0030 PRINT "DEGREE:",R
0040 PRINT
0050 PRINT "ENTER ROOTS: x+iy IS ENTERED AS x,y"
0060 PRINT
0070 PRINT "INDEX","REAL PART","IMAGINARY PART"
0080 PRINT
0090 FOR I=1 TO R STEP 1
0100 DISP "ENTER x";I;",y";I;
0110 INPUT E(I),F(I)
0120 PRINT I,E(I),F(I)
0130 NEXT I
0140 LET I=FNA(R)
0150 PRINT
0160 PRINT "COEFFICIENTS:"
0170 PRINT
0180 PRINT "INDEX","REAL PART","IMAGINARY PART"
0190 PRINT
0200 FOR I=1 TO R+1 STEP 1
0210 PRINT I-1,A(I),B(I)
0220 NEXT I
0230 PRINT
0240 PRINT

END OF LISTING


LINK *SLCOCC,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
DEGREE:           5

ENTER ROOTS: x+iy IS ENTERED AS x,y

INDEX             REAL PART       IMAGINARY PART

 1                 1                1
 2                 2                2
 3                 3                3
 4                 4                4
 5                 5                5

COEFFICIENTS:

INDEX             REAL PART       IMAGINARY PART

 0                 480              480
 1                -1096             0
 2                 450             -450
 3                 0                170
 4                -15              -15
 5                 1                0
```

*SLEMF

Title Complete Elliptic Integral of Second Kind

Purpose Evaluation of the Complete Elliptic Integral of Second Kind

$$* \text{SLEMF} \qquad E(x) = \int_0^{\pi/2} \sqrt{1 - x \sin^2 \vartheta} \; d\vartheta$$

Method The value is computed by means of the representation

$$E(x) = \frac{1}{2}\pi \cdot \left\{ 1 - \left(\frac{1}{2}\right)^2 \frac{x}{1} - \left(\frac{1 \cdot 3}{2 \cdot 4}\right)^2 \frac{x^2}{2} - \left(\frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6}\right)^2 \frac{x^3}{3} - \ldots\ldots \right\}$$

Calling Statement PRINT FNA (X)

Parameter X Argument

Global variables None

Listing

```
OLD *SLEMF
LIST
FILE    *SLEMF

0010 DEF FNA(X)K,S,Q
0020 LET S=Q=1
0030 FOR K=1 TO 1000 STEP 1
0040 LET Q=Q*X*((2*K-1)/(2*K))↑2
0050 IF ABS(Q)<1E-13 THEN 80
0060 LET S=S-Q/(2*K-1)
0070 NEXT K
0080 LET S=S*PI/2
0090 LET FN*=S
0100 FNEND

END OF LISTING
```

Example of use

```
OLD +EMF
LIST
FILE     +EMF


0010 !              ##.##        ##.############
0020 PRINT
0030 PRINT TAB(5);"COMPLETE ELLIPTIC INTEGRAL OF SECOND KIND: E(X)"
0040 REM\\\\\\COMPLETE ELLIPTIC INTEGRAL OF THE SECOND KIND E(X)\\\\\\\\\\\\
0050 PRINT
0060 PRINT ,"X","E(X)"
0070 PRINT
0080 DISP "INPUT X";
0090 INPUT X
0100 PRINT USING 10,X,FNA(X)
0110 GOTO 80

END OF LISTING


LINK *SLEMF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

    COMPLETE ELLIPTIC INTEGRAL OF SECOND KIND: E(X)

                X                E(X)

             0.00         1.570796326795
             0.10         1.530757636897
             0.20         1.489035058094
             0.30         1.445363064411
             0.40         1.399392138896
             0.50         1.350643881046
             0.60         1.298428035044
             0.70         1.241670567943
             0.80         1.178489924323
             0.90         1.104774732695
             1.00         1.000249306212
```

*SLKMF

Title Complete Elliptic Integral of First Kind

Purpose Evaluation of the Complete Elliptic Integral of First Kind

*SLKMF $$K(x) = \int_0^{\pi/2} \frac{d\vartheta}{\sqrt{1 - x \sin^2\vartheta}}$$

Method The value is computed by means of the representation

$$k(x) = \frac{1}{2}\pi\left[1+\left(\frac{1}{2}\right)^2 x + \left(\frac{1\cdot3}{2\cdot4}\right)^2 x^2 + \left(\frac{1\cdot3\cdot5}{2\cdot4\cdot6}\right)^2 x^3 + \ldots.\right]$$

Calling Statement PRINT FNA (X)

Parameter X Argument

Global variables None

Listing

```
OLD *SLKMF
LIST
FILE     *SLKMF

0010 DEF FNA(X)S,Q,K
0020 LET S=Q=1
0030 FOR K=1 TO 1000 STEP 1
0040 LET Q=Q*X*((2*K-1)/(2*K))↑2
0050 IF ABS(Q)<1E-15 THEN 80
0060 LET S=S+Q
0070 NEXT K
0080 LET S=S*PI/2
0090 LET FN*=S
0100 FNEND

END OF LISTING
```

Example of use

```
OLD *KMF
LIST
FILE     *KMF

0010 !              ##.##        ##.############
0020 PRINT
0030 PRINT TAB(5),"COMPLETE ELLIPTIC INTEGRAL OF FIRST KIND: K(X)"
0040 REM\\\\\\COMPLETE ELLIPTIC INTEGRAL OF THE FIRST KIND K(X)\\\\\\\\\\
0050 PRINT
0060 PRINT ,"X","K(X)"
0070 PRINT
0080 DISP "INPUT X";
0090 INPUT X
0100 PRINT USING 10,X,FNA(X)
0110 GOTO 80

END OF LISTING


LINK *SLKMF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

    COMPLETE ELLIPTIC INTEGRAL OF FIRST KIND: K(X)

                X                K(X)

             0.00         1.570796326795
             0.10         1.612441348713
             0.20         1.659623598588
             0.30         1.713889448163
             0.40         1.777519371468
             0.50         1.854074677271
             0.60         1.949567749772
             0.70         2.075363135239
             0.80         2.257205326743
             0.90         2.578092113176
             1.00         5.129154720175
```

*SLLAGG

Title Generalized Laguerre Polynomial

$L_n^{(a)}(x)$

Purpose Evaluation of the Generalized Laguerre Polynomial

$L_n^{(a)}(x)$

Method The value is computed by means of the recurrence relation

$$L_0^{(a)}(x) = 1 \quad , \quad L_1^{(a)}(x) = a + 1 - x$$

$$n\, L_n^{(a)}(x) = (2n + a - 1 - x) \cdot L_{n-1}^{(a)}(x) - (a + n - 1) \cdot L_{n-2}^{(a)}(x)$$

Calling Statement *PRINT FNA (N, A, X)

Parameter

N Degree

A See above [weight function $e^{-x}\, x^{A}$

X Argument

Global variables None

Listing

```
OLD *SLLAGG
LIST
FILE    *SLLAGG


0010 DEF FNA(N,A,X)K,L0,L1
0020 LET L=L0=1
0030 LET L1=A+1-X
0040 FOR K=2 TO N STEP 1
0050 LET L=((2*K+A-1-X)*L1-(A+K-1)*L0)/K
0060 LET L0=L1
0070 LET L1=L
0080 NEXT K
0090 IF N=1 THEN 120
0100 LET FN*=L
0110 GOTO 130
0120 LET FN*=L1
0130 FNEND

END OF LISTING
```

Example of use

```
OLD +LAGG
LIST
FILE    +LAGG

0010 DCL 40A$
0020 PRINT
0030 PRINT "GENERALIZED LAGUERRE POLYNOMIAL  LA(N](X)"
0040 PRINT
0050 DISP "ENTER A,N,X";
0060 INPUT A,N,X
0070 PRINT
0080 BUILD A$,"A=",A," LC",N,"](",X,")="
0090 PRINT USING 100,A$,FNA(N,A,X)
0100 :         'RRRRRRRRRRRRRRRRRRRRRRRR####.##############
0110 GOTO 50

END OF LISTING


LINK *SLLAGG,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

GENERALIZED LAGUERRE POLYNOMIAL  LA(N](X)

              A= 0   LC 0 ](  .5 )=   1.00000000000000

              A= 0   LC 1 ](  .5 )=   0.50000000000000

             A= 0   LC 10 ](  .5 )=  -0.38937441413650

               A= 0   LC 5 ](  5 )=  -3.16666666666400

             A= 0   LC 10 ](  10 )=  27.98412698409000

              A= 1   LC 0 ](  .5 )=   1.00000000000000

               A= 1   LC 5 ](  5 )=  -1.29166666666700

             A= 1   LC 10 ](  10 )=  14.79188712521000

               A= 5   LC 5 ](  5 )=  -1.12499999999800
```

*SLHNF

Title Hermite Polynomial H_n (X)

Purpose Evaluation of the Hermite Polynomial H_n (X)

Method The value is computed by means of the recurrence relation

$$H_0(x) = 1, \quad H_1(x) = 2x$$

$$H_n(x) = 2x\,H_{n-1}(x) - 2(n-1)\,H_{n-2}(x)$$

Calling Statement PRINT FNA (N, X)

Parameter
- N Degree
- X Argument

Global variables None

Listing

```
OLD *SLHNF
LIST
FILE     *SLHNF

0010 DEF FNA(N,X)K,H0,H1
0020 LET H=H0=1
0030 LET H1=2*X
0040 FOR K=2 TO N STEP 1
0050 LET H=2*(X*H1-(K-1)*H0)
0060 LET H0=H1
0070 LET H1=H
0080 NEXT K
0090 IF N=1 THEN 120
0100 LET FN*=H
0110 GOTO 130
0120 LET FN*=H1
0130 FNEND

END OF LISTING
```

Example of use

```
FILE    *HNF

0010 DCL 40A#
0020 PRINT
0030 PRINT TAB(15);"HERMITE POLYNOMIALS  HEN(X)"
0040 PRINT
0050 DISP "INPUT N,X";
0060 INPUT N,X
0070 BUILD A#,"HE",N,"(",X,")="
0080 :    'RRRRRRRRRRRRRRRR######.##############
0090 PRINT USING 80:A#,FNA(N,X)
0100 GOTO 50

END OF LISTING


LINK *SLHNF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

               HERMITE POLYNOMIALS  HEN(X)

          HE  0 )( 0 )=      1.00000000000000
          HE  1 )( 1 )=      2.00000000000000
          HE  2 )( 2 )=     14.00000000000000
          HE  2 )( 3 )=    160.00000000000000
         HE  2 )( 10 )=    396.00000000000000
          HE  3 )( 5 )=   3440.00000000000000
          HE  3 )( 1 )= -1648.00000000000000
        HE 10 )( .5 )= 22991.00000000000000
```

*SLHEN

Title — Hermite Polynomial $He_n(x)$

Purpose — Evaluation of the Hermite Polynomial $He_n(x)$

Method — The value is computed be means of the recurrence relation

$$He_0(x) = 1 \quad , \quad He_1(x) = x$$

$$He_n(x) = x\,He_{n-1}(x) - (n-1)\,He_{n-2}(x)$$

Calling Statement — PRINT FNA (N, X)

Parameter

N Degree

X Argument

Global variables — None

Listing

```
OLD *SLHEN
LIST
FILE    *SLHEN

0010 DEF FNA(N,X)K,H0,H1,H
0020 LET H=H0=1
0030 LET H1=X
0040 FOR K=2 TO N STEP 1
0050 LET H=X*H1-(K-1)*H0
0060 LET H0=H1
0070 LET H1=H
0080 NEXT K
0090 IF N=1 THEN 120
0100 LET FN*=H
0110 GOTO 130
0120 LET FN*=H1
0130 FNEND

END OF LISTING
```

Example of use

```
OLD +HEN
LIST
FILE     +HEN


0010 DCL 40A$
0020 PRINT
0030 PRINT TAB(10);"HERMITE POLYNOMIAL HE(N)(X)"
0040 PRINT
0050 DISP "INPUT N;X";
0060 INPUT N,X
0070 BUILD A$,"HE(",N,")(",X,")="
0080 :        'RRRRRRRRRRRRRRRR#####.##############
0090 PRINT USING 80,A$,FNA(N,X)
0100 PRINT
0110 GOTO 50

END OF LISTING


LINK *SLHEN,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

          HERMITE POLYNOMIAL HE(N)(X)

             HE( 0 )( 0 )=     1.000000000000

             HE( 1 )( 1 )=     1.000000000000

             HE( 2 )( 2 )=     3.000000000000

             HE( 3 )( 1 )=     6.000000000000

             HE( 5 )( 5 )= 1950.000000000000

            HE( 10 )( .5 )=   49.043945312500

             HE( 10 )( 1 )= 1216.000000000000
```

*SLFOUR

Title Evaluation of Fourier Series

Purpose Evaluate a Fourier formula

Method The following expression is used for calculation

$$F(x) = \frac{A_0}{2} + \sum_{k=1}^{N} (A_k \cos Kx + B_K \sin Kx)$$

Calling Statement F = FNZ (N, X)

Parameters

N Maximum harmonic order to be considered

X Value for which the serie will be evaluated

Global variables

A ( ) Array of A values

B ( ) Array of B values

Remember that for N greater than 9 a DIM Statement must be placed in the Calling program

Return The value of the function

Listing

```
OLD *SLFOUR
LIST
FILE    *SLFOUR

0010 DEF FNA(N,X)I,K,S
0100 LET S=A(1)/2
0110 FOR K=1 TO N STEP 1
0120 LET S=S+A(K+1)*COS(K*X)+B(K)*SIN(K*X)
0130 NEXT K
0140 LET FN*=S
0150 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE    +FOUR

0010 DIM A(100),B(100)
0020 PRINT
0030 PRINT TAB(10);"EVALUATION OF FOURIER SERIES"
0040 PRINT
0050 DISP "INPUT N,X";
0060 INPUT N,X
0070 PRINT "COEFFICIENTS:"
0080 FOR I=N+1 TO 1 STEP -1
0090 DISP "INPUT A";I-1;
0100 INPUT A(I)
0110 PRINT "A(";I-1;")=";A(I),
0120 NEXT I
0130 PRINT
0140 FOR I=N TO 1 STEP -1
0150 DISP "INPUT B";I-1;
0160 INPUT B(I)
0170 PRINT "B(";I-1;")=";B(I),
0180 NEXT I
0190 PRINT
0200 PRINT
0210 PRINT "RESULT: F(";X;")=",FNA(N,X)
0220 PRINT
0230 GOTO 50

END OF LISTING


LINK *SLFOUR,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

         EVALUATION OF FOURIER SERIES

COEFFICIENTS:
A( 5 )= 1        A( 4 )= 2        A( 3 )= 3        A( 2 )= 4        A( 1 )= 5
A( 0 )= 6
B( 4 )= 1        B( 3 )= 2        B( 2 )= 3        B( 1 )= 4        B( 0 )= 5


RESULT: F( 1 )=  5.8386970

COEFFICIENTS:
A( 10 )= 1       A( 9 )= 2        A( 8 )= 3        A( 7 )= 4        A( 6 )= 5
A( 5 )= 6        A( 4 )= 7        A( 3 )= 8        A( 2 )= 9        A( 1 )= 0
A( 0 )= 10
B( 9 )= 1        B( 8 )= 2        B( 7 )= 3        B( 6 )= 4        B( 5 )= 5
B( 4 )= 6        B( 3 )= 7        B( 2 )= 8        B( 1 )= 9        B( 0 )= 0


RESULT: F( 2 )= -1.1907437
```

*SLIGAM

Title. Incomplete Gamma Function (a, x)

Purpose Calculate the values of the function

* SLIGAM

$$\gamma(a,x) = P(a,x)\,\Gamma(a) = \int_0^x e^{-t}\,t^{a-1}\,dt \qquad (a>0)$$

Method A series approximation is used

$$\gamma(a,x) = x^a \sum_{n=0}^{\infty} \frac{(-x)^n}{(a+n)n!}$$

Calling Statement F = FNZ (A, X)

Parameters

A =
X = } Values for which the function will be calculated

Global variables None

Return The value of the function

Listing

```
OLD *SLIGAM
LIST
FILE     *SLIGAM


0010 DEF FNA(A,X)K,Q,S
0020 LET S=Q=1/A
0030 FOR K=1 TO 69 STEP 1
0040 LET Q=-Q*X*(A+K-1)/(K*(K+A))
0050 IF ABS(Q)<1E-15 THEN 80
0060 LET S=S+Q
0070 NEXT K
0080 LET S=S*X↑A
0090 LET FN*=S
0100 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE    +IGAM

0010 PRINT
0020 PRINT TAB(18);"INCOMPLETE GAMMA FUNCTION "
0030 PRINT
0040 PRINT ,"A","X","VALUE"
0050 PRINT
0060 DISP "INPUT A,X";
0070 INPUT A,X
0080 PRINT ,A,X,FNA(A,X)
0090 PRINT
0100 GOTO 60

END OF LISTING


LINK *SLIGAM,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
```

INCOMPLETE GAMMA FUNCTION

A	X	VALUE
1	1	.63212056
1	5	.99326205
1	10	.99995460
2	1	.26424112
2	5	.95957232
2	10	.99950060
5	1	8.7836324E-02
5	5	13.428161
5	10	23.297940

*SLGAMA

Title Gamma Function

Purpose Calculate the function

*SLGAMA

$$\gamma(x) = \int_0^\infty t^{x-1}\, e^{-x}\, dt \qquad (x>0)$$

Method For X in the range of $0 \leqslant X \leqslant 1$ an approximating polynomial is used for calculation $\gamma(1+x)$ Outside os the interval the following recurrence formulaes are used

$$\gamma(x+1) = x\,\gamma(x)$$

$$\gamma(n+x) = (n-1+x)(n-2+x) \ldots\ldots (1+x)\,\gamma(1+x)$$

Calling Statement F = FNZ (X)

Parameter X = value for which the function will be calculated

Global variables None

Return The value of the function

Listing

```
OLD *SLGAMA
LIST
FILE    *SLGAMA

0010 DEF FNA(X1)X,T,K
0020 LET X=X1-INT(X1)
0030 LET T=X*(X*(X*(X*(0.035868343*X-0.193527818)+0.482199394)-0.756704078)
0040 LET T=X*(X*(X*(X*(X*(T+0.918206857)-0.897056937)+0.988205891)-0.577191652)+1
0050 FOR K=INT(X1)-1 TO 1 STEP -1
0060 LET T=T*(K+X)
0070 NEXT K
0080 IF X1>=1 THEN 100
0090 LET T=T/X1
0100 LET FN*=T
0110 FNEND

END OF LISTING
```

Example of use

```
OLD TEST
LIST
FILE     TEST


0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LINK *SLGAMA,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X              F(X)
 1               1
 .25             3.6256107
 10              362880
 20              1.2164510E+17
 30              8.8417620E+30
 40              2.0397882E+46
 50              6.0828186E+62
```

*SLBER

Title — Kelvin Function ber (x) of zero order

Purpose — The function is defined by the relation

$$b_{er}(x) + b_{ei}(x) = J_0\left(x\, e^{\frac{3\pi i}{4}}\right)$$

The function calculates ber (x)

Method — It is used a series approximation

$$b_{er}(x) = 1 - \frac{u^2}{(2!)^2} + \frac{u^4}{(4!)^2} - \frac{u^6}{(6!)^2} + \cdots\cdots$$

where

$$u = \frac{1}{4}x^2$$

Calling Statement — F = FNZ (X)

Parameter — X = value for which the function will be calculated

Global variables — None

Return — The value of the function

Listing

```
OLD *SLBER
LIST
FILE    *SLBER

0010 DEF FNA(X)K,Q,S
0020 LET Q=S=1
0030 FOR K=1 TO 1000 STEP 1
0040 LET Q=Q*X*X/(K*K*4)
0050 LET S=S+Q*COS(PI*K/2)
0060 IF ABS(Q)<1E-13 THEN 80
0070 NEXT K
0080 LET FN*=S
0090 FNEND

END OF LISTING
```

Example of use

```
OLD *BER
LIST
FILE    *BER

0010 DCL 30A$
0020 PRINT
0030 PRINT TAB(28);"KELVIN FUNCTION  ber(X)"
0040 PRINT
0050 DISP "INPUT X";
0060 INPUT X
0070 :=######.##############
0080 BUILD USING 70,A$,FNA(X)
0090 PRINT TAB(20),"ber(";X;")",A$
0100 PRINT
0110 GOTO 50

END OF LISTING


LINK *SLBER,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
                          KELVIN FUNCTION  ber(X)

                  ber( 0 )      =      1.000000000000

                  ber( .1 )     =      0.9999984375000

                  ber( .5 )     =      0.9990234639907

                  ber( 1 )      =      0.9843817812130

                  ber( 2 )      =      0.7517341827138

                  ber( 2.5 )    =      0.3999684171294

                  ber( 3 )      =     -0.2213802495985

                  ber( 5 )      =     -6.2300824786640
```

*SLBEI

Title Kelvin Function bei (x) of zero order

Purpose The function is defined by the relation

$$bei\,(x) - ber\,(x) = J_0\left(x\, e^{\frac{3\pi i}{4}}\right)$$

The routine calculates the values of bei (x)

Method An approximation series is used

$$bei\,(x) = u - \frac{u^3}{(3!)^2} + \frac{u^5}{(5!)^2} - \ldots\ldots$$

where

$$u = \frac{1}{4}x^2$$

Calling Statement F = FNZ (X)

Parameter X = Value for which the function will be calculated

Global variables None

Return The value of the function

Listing

```
FILE    *SLBEI

0010 DEF FNR(X)K,Q,S
0020 LET Q=1
0030 LET S=0
0040 FOR K=1 TO 1000 STEP 1
0050 LET Q=Q*X*X/(K*K*4)
0060 LET S=S+Q*SIN(PI*K/2)
0070 IF ABS(Q)<1E-13 THEN 90
0080 NEXT K
0090 LET FN*=S
0100 FNEND

END OF LISTING
```

Example of use

```
OLD *BEI
LIS
FILE    *BEI


0010 DCL 30A$
0020 PRINT
0030 PRINT TAB(28);"KELVIN FUNCTION  bei(X)"
0040 PRINT
0050 DISP "INPUT X";
0060 INPUT X
0070 :=#######.##############
0080 BUILD USING 70,A$,FNA(X)
0090 PRINT TAB(20);"bei(";X;")",A$
0100 PRINT
0110 GOTO 50

END OF LISTING


LINK *SLBEI,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

                          KELVIN FUNCTION  bei(X)

                bei( 0 )     =     0.0000000000000

                bei( .1 )    =     0.0024999995660

                bei( .2 )    =     0.0099999722222

                bei( .5 )    =     0.0624932183822

                bei( 1 )     =     0.2495660400366

                bei( 2 )     =     0.9722916273066

                bei( 3.5 )   =     2.2832499668520

                bei( 5 )     =     0.1160343815505
```

*SLERF

Title — Error Function erf (x)

Purpose — Calculate the value of the function

$$Erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

Method — An approximation serie is used

$$erf(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}$$

Calling Statement — F = FNZ (X)

Parameter — x = Value for which the function will be calculated

Global variables — None

Return — The value of the function

Listing

```
OLD *SLERF
LIST
FILE    *SLERF

0010 DEF FNA(X)K,Q,S
0020 LET Q=S=X
0030 FOR K=1 TO1000 STEP 1
0040 LET Q=-Q*X*X*(K+K-1)/((K+K+1)*K)
0050 IF ABS(Q)<1E-15 THEN 80
0060 LET S=S+Q
0070 NEXT K
0080 LET S=S*2/SQR(PI)
0090 LET FN*=S
0100 FNEND

END OF LISTING
```

Example of use

```
OLD TEST
LIST
FILE     TEST


0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LINK *SLERF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X               F(X)
 .1               .11246292
 .15              .16799597
 .2               .22270259
 .5               .52049988
 2                .99532227
```

*SLBJN

Title Bessel function of Integer order $J_n(x)$

Purpose $J_n(x)$ is a solution of the differential equation

$$x^2 \frac{d^2}{dx^2} y + x \frac{dy}{dx} + (x^2 - n^2) y = 0$$

The routine evaluates the function y

Method The value is computed by means of the series representation

$$J_n(x) = \left(\frac{1}{2} x\right)^n \sum_{K=0}^{\infty} \frac{\left(-\frac{1}{4} x^2\right)^K}{K!\,\Gamma(n+K+1)}$$

and the recurrence relation

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x)$$

Calling Statement PRINT FNA (N,X)

Parameter

N Order

X Argument

Global variables None

Listing

```
OLD *SLBJN
LIST
FILE    *SLBJN

0010 DEF FNZ(N,X)Q,Q1,S,S1,K,Q2,I,J,J1,J0
0020 LET S1=0
0040 LET Q=Q1=S=1
0050 LET Q2=X*X/4
0060 FOR K=1 TO 69 STEP 1
0070 LET Q=-Q*Q2/(K*K)
0080 IF ABS(Q)<1E-15 THEN 130
0090 LET Q1=-Q1*Q2/(K*K+K)
0100 LET S=S+Q
0110 LET S1=S1+Q1
0120 NEXT K
0130 LET J=J0=S
0140 LET J1=S1*X/2+X/2
0150 FOR I=2 TO N STEP 1
```

```
0160 LET J=2*(I-1)/X*J1-J0
0170 LET J0=J1
0180 LET J1=J
0190 NEXT I
0200 IF N=1 THEN 230
0210 LET FN*=J
0220 GOTO 240
0230 LET FN*=J1
0240 FNEND

END OF LISTING
```

Example of use

```
OLD +BJN
LIST
FILE    +BJN


0010 :  ####          ####.##     ######.#############
0020 PRINT
0030 PRINT "BESSEL FUNCTION J[N](X)"
0050 PRINT
0060 PRINT "ORDER N","X","J[N](X)"
0070 PRINT
0080 DISP "INPUT N,X";
0090 INPUT N,X
0100 PRINT USING 10,N,X,FNA(N,X)
0110 GOTO 80

END OF LISTING


LINK *SLBJN,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

BESSEL FUNCTION J[N](X)

ORDER N          X                J[N](X)

     0            0.10          0.997501562066
     0            0.20          0.990024972240
     0            0.50          0.938469807241
     0            1.00          0.765197686558
     0            1.50          0.511827671736
     0            2.00          0.223890779141
     0            2.50         -0.048383776468
     1            0.10          0.049937526036
     1            1.00          0.440050585745
     1            2.00          0.576724807757
     1            5.00         -0.327579137582
     2            0.10          0.001248958659
     2            0.20          0.004983354153
     2            1.00          0.114903484932
     2            2.00          0.352834028616
     2            5.00          0.046565116281
     3            0.10          0.000020820320
     3            0.50          0.002563729994
     3            1.00          0.019563353984
     3            2.00          0.128943249475
     3            5.00          0.364831230607
     4            0.10          0.000000260527
     4            4.00          0.281129064951
     4            1.00          0.002476638971
     4            2.00          0.033995719810
     4            5.00          0.391232360447
     5            0.10          0.000000021816
     5            1.00          0.000249757781
     5            2.00          0.007039529763
     5            5.00          0.261140546109
     5           10.00         -0.234061527891
```

*SLBES1

Title Spherical Bessel-function of first kind $j_n(x)$

Purpose Evaluation of the spherical Bessel-function of first kind of order

Method The value is composed by means of the representation

$$j_n(x) = \frac{x^n}{1\cdot 3\cdot 5\cdot \ldots \cdot (2n+1)} \cdot \left\{ 1 - \frac{\frac{1}{2}x^2}{1!\,(2n+3)} + \frac{\left(\frac{1}{2}x^2\right)^2}{2!\,(2n+3)(2n+5)} - \ldots \right\}$$

Calling Statement PRINT FNA (N, X)

Parameter

N Order

X Argument

Global variables None

Listing

```
LIST
FILE    *SLBES1

0010 DEF FNA(N,X)D,I,K,S,Q,U
0020 LET D=S=Q=1
0030 LET U=X*X/2
0040 FOR I=1 TO 2*N+1 STEP 2
0050 LET D=D/I
0060 NEXT I
0070 FOR K=1 TO 1000 STEP 1
0080 LET Q=-Q*U/(K*(2*N+2*K+1))
0090 IF ABS(Q)<1E-15 THEN 120
0100 LET S=S+Q
0110 NEXT K
0120 LET S=X↑N*S*D
0130 LET FN*=S
0140 FNEND

END OF LISTING
```

Example of use

```
OLD +BES1
LIST
FILE    +BES1

0010 !                ####         ####.##         ##.############
0020 PRINT
0030 PRINT TAB(15);"SPHERICAL BESSEL FUNCTION OF THE FIRST KIND"
0050 PRINT
0060 PRINT ,"ORDER N","X","jEN](X)"
0070 PRINT
0080 DISP "INPUT N,X";
0090 INPUT N,X
0100 PRINT USING 10,N,X,FNA(N,X)
0110 GOTO 80

END OF LISTING


LINK *SLBES1,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
```

SPHERICAL BESSEL FUNCTION OF THE FIRST KIND

ORDER N	X	jEN](X)
0	1.00	0.841470984808
0	5.00	-0.191784854932
0	10.00	-0.054402111005
1	1.00	0.301168678940
1	5.00	-0.095089408079
1	10.00	0.078466941812
5	10.00	-0.055534511622
7	1.00	0.000080479013
7	5.00	0.017902778178
7	10.00	0.113386230655

*SLBES2

Title Spherical Besselfunction of second kind $y_n(x)$

Purpose Evaluation of spherical Bessel-function of second kind of order

Method The value is computed by means of the representation

$$y_n(x) = \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1)}{x^{n+1}} \cdot \left\{ 1 - \frac{\frac{1}{2}x^2}{1!(1-2n)} + \frac{(\frac{1}{2}x^2)^2}{2!(1-2n)(3-2n)} - \ldots \right\}$$

Calling Statement PRINT FNA (N, X)

Parameter
N Order
X Argument

Global variables None

Listing

```
LIST
FILE    *SLBES2

0010 DEF FNA(N,X)D,I,K,S,Q,U
0020 LET D=S=Q=1
0030 LET U=X*X/2
0040 FOR I=1 TO 2*N-1 STEP 2
0050 LET D=D*I
0060 NEXT I
0070 FOR K=1 TO 1000 STEP 1
0080 LET Q=-Q*U/(K*(2*(K-N)-1))
0090 IF ABS(Q)<1E-15 THEN 120
0100 LET S=S+Q
0110 NEXT K
0120 LET S=-S*D/X↑(N+1)
0130 LET FN*=S
0140 FNEND

END OF LISTING
```

Example of use

```
OLD +BES2
LIST
FILE    +BES2

0010 :                ####          ####.## #########.############
0020 PRINT
0030 PRINT TAB(15);"BESSEL FUNCTION J[N](X)"
0050 PRINT
0060 PRINT ,"ORDER N","X","J[N](X)"
0070 PRINT
0080 DISP "INPUT N,X";
0090 INPUT N,X
0100 PRINT USING 10,N,X,FNA(N,X)
0110 GOTO 80

END OF LISTING


LINK *SLBES2,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

              BESSEL FUNCTION J[N](X)

                ORDER N          X              J[N](X)

                  0            10.00          0.083907152771
                  0             5.00         -0.056732437093
                  0             1.00         -0.540302305868
                  1             1.00         -1.381773290674
                  1             5.00          0.180438367515
                  1            10.00          0.062792826361
                  5             1.00       -999.440343388900
                  5             5.00         -0.320465046749
                  5            10.00          0.093833541693
```

*SLSINX

Title Modified Spherical Bessel-function of first kind $i_n(x)$

Purpose Evaluation of the Modified Besselfunction of first kind of order

Method The value is computed by means of the representation

$$i_n(x) = \frac{x^n}{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n+1)} \cdot \left\{ 1 + \frac{\frac{1}{2}x^2}{1!\,(2n+3)} + \frac{\left(\frac{1}{2}x^2\right)^2}{2!\,(2n+3)(2n+5)} + \ldots \right\}$$

Calling Statement PRINT FNA (N, X)

Parameter

N Order

X Argument

Global variables None

Listing

```
OLD *SLINX
LIST
FILE    *SLINX

0010 DEF FNA(N,X)D,I,K,Q,S,U
0020 LET D=S=Q=1
0030 LET U=X*X/2
0040 FOR I=1 TO 2*N+1 STEP 2
0050 LET D=D/I
0060 NEXT I
0070 FOR K=1 TO 1000 STEP 1
0080 LET Q=Q*U/(K*(2*N+2*K+1))
0090 IF ABS(Q)<1E-15 THEN 120
0100 LET S=S+Q
0110 NEXT K
0120 LET S=X↑N*S*D
0130 LET FN*=S
0140 FNEND

END OF LISTING
```

Example of use

```
OLD +SINX
ERROR  187
OLD +INX
LIST
FILE    +INX

0010 :                ####          ####.##     ######.#############
0020 PRINT
0030 PRINT TAB(15);"MODIFIED SPHERICAL BESSEL FUNCTION OF THE FIRST KIND"
0050 PRINT
0060 PRINT ,"ORDER N","X","i[N](X)"
0070 PRINT
0080 DISP "INPUT N,X";
0090 INPUT N,X
0100 PRINT USING 10,N,X,FNA(N,X)
0110 GOTO 80

END OF LISTING


LINK *SLINX,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

              MODIFIED SPHERICAL BESSEL FUNCTION OF THE FIRST KIND

   ORDER N          X                i[N](X)

      0             0.00           1.000000000000
      0             1.00           1.175201193640
      0             2.00           1.813430293920
      0             5.00          14.846642115500
      0            10.00        1101.323287462000
      1             0.00           0.000000000000
      1             5.00          11.873861281820
      1            10.00         991.190963259500
      5             0.00           0.000000000000
      5             2.00           0.003584848301
      5             1.00           0.000093962375
      5            10.00         236.839582705100
```

*SLII0X

Title — Definite Integral of the Bessel function I_o (x)

Purpose — Calculate the values of the integral

$$f(x) = \int_o^x I_o(t)\ dt$$

Method — A serie approximation is used

$$f(x) = x + \sum_{n=1}^{\infty} \frac{x}{(n!)^2(2n+1)}$$

Calling Statement — F = FNZ (X)

Parameters — X = superior limit of integration

Global variables — None

Return — The value of the function

Listing

```
OLD *SLII0X
LIST
FILE    *SLII0X

0010 DEF FNA(X)K,S,Q,U
0020 LET S=Q=1
0030 LET U=X*X/4
0040 FOR K=1 TO 40 STEP 1
0050 LET Q=Q*U*(2*K-1)/(K*K*(2*K+1))
0060 IF ABS(Q)<1E-14 THEN 90
0070 LET S=S+Q
0080 NEXT K
0090 LET S=X*S
0100 LET FN*=S
0110 FNEND

END OF LISTING
```

Example of use

```
0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LIN* *SLII0X,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X               F(X)
 0                0
 .1               .10008336
 .5               .51051481
 .7               .72911369
 .9               .96262523
 1                1.0865211
 3                6.1609615
 5                31.848668
```

Title Fresnel Integral S (x)

Purpose Calculate the values of the integral

$$S(x) = \int_0^x \sin\left(\frac{\pi}{2} t^2\right) dt$$

Method An approximation serie is used

$$S(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \left(\frac{\pi}{2}\right)^{2n+1}}{(2n+1)!\,(4n+3)} x^{4n+3} \qquad \text{for } x < 3.5$$

Calling Statement F = FNZ (X)

Parameter X = upper limit of integration

Global variables None

Return The value of the function

Listing

```
OLD *SLSF
LIST
FILE    *SLSF

0020 DEF FNA(X)K,S,Q,U,Q1
0030 LET K=1
0040 LET S=X*X*X*PI/6
0050 LET Q=S*3
0060 LET U=Q*X*PI/2
0070 LET Q=-Q*U/(2*K*(2*K+1))
0080 LET Q1=Q/(4*K+3)
0090 IF ABS(Q1)<1E-15 THEN 130
0100 LET S=S+Q1
0110 LET K=K+1
0120 GOTO 70
0130 LET FN*=S
0140 FNEND

END OF LISTING
```

Example of use

```
OLD TEST
LIST
FILE     TEST


0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LINK *SLSF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X               F(X)
 0                0
 .1               5.2358955E-04
 .5               6.4732433E-02
 1                .43825915
 1.1              .53649791
 2.5              .61918176
 3.1              .58181585
```

*SLCF

Title Fresnel Integral C (x)

Purpose Calculate the integral

$$C(x) = \int_0^x \cos\left(\frac{\pi}{2} t^2\right) \, dt$$

Method A series approximation is used

$$C(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \left(\frac{\pi}{2}\right)^{2n}}{(2n)!\,(4n+1)} x^{4n+1} \qquad \text{for } x < 3.5$$

Calling Statement F = FNZ (X)

Parameters X = superior limit of integration

Global variables None

Return The value of the function

Listing

```
OLD *SLCF
LIST
FILE    *SLCF

0020 DEF FNA(X)F,K,U,S,Q,Q1
0030 LET K=F=1
0040 LET S=Q=X
0050 LET U=X*X*X*X*PI*PI/4
0060 LET Q=-Q*U/((K+K)*(K+K-1))
0070 LET Q1=Q/(4*K+1)
0080 IF ABS(Q1)<1E-14 THEN 120
0090 LET S=S+Q1
0100 LET K=K+1
0110 GOTO 60
0120 LET FN*=S
0130 FNEND

END OF LISTING
```

Example of use

```
OLD TEST
LIST
FILE     TEST


0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LINK *SLCF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X               F(X)
 0                0
 1                .77989340
 1.1              .76380667
 2.5              .45741301
 3                .60572079
 3.1              .56159390
```

*SLLEG1

Title — Associated Legendre Function of First Kind $P_n(x)$

Purpose — Evaluation of the Legendre Function of first kind and arbitrary degree

Method — The value is computed by means of the recursion formula

$$P_0(x) = 1 \quad , \quad P_1(x) = x$$

$$n\,P_n(x) = (2n-1)\cdot x\cdot P_{n-1}(x) - (n-1)\,P_{n-2}(x)$$

Calling Statement — PRINT FNA (N, X)

Parameter

N Degree

X Argument

Global variables — None

Listing

```
OLD *SLLEG1
LIST
FILE     *SLLEG1

0010 DEF FNA(N,X)K,P0,P1,P
0020 LET P=P0=1
0030 LET P1=X
0040 FOR K=2 TO N STEP 1
0050 LET P=((K+K-1)*X*P1-(K-1)*P0)/K
0060 LET P0=P1
0070 LET P1=P
0080 NEXT K
0090 IF N=1 THEN 120
0100 LET FN*=P
0110 GOTO 130
0120 LET FN*=P1
0130 FNEND

END OF LISTING
```

Example of use

```
OLD +LEG1
LIST
FILE    +LEG1

0010              ####        ####.##    ######.############
0020 PRINT
0030 PRINT TAB(20);"LEGENDRE FUNCTION OF FIRST KIND"
0040 PRINT
0050 PRINT ,"DEGREE N","X","P[N](X)"
0060 PRINT
0070 DISP "INPUT N,X";
0080 INPUT N,X
0090 PRINT USING 10,N,X,FNA(N,X)
0100 GOTO 70

END OF LISTING


LINK *SLLEG1.A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
```

LEGENDRE FUNCTION OF FIRST KIND

DEGREE N	X	P[N](X)
0	0.00	1.000000000000
1	1.00	1.000000000000
2	0.00	-0.500000000000
2	0.50	-0.125000000000
2	0.71	0.256150000000
3	0.45	-0.447187500000
3	0.49	-0.440877500000
5	0.25	0.339721679687
5	0.50	0.089843750000
10	0.00	-0.246093750000
10	0.50	-0.188228607178
10	0.75	0.264374513178
10	1.00	1.000000000000

*SLLEG2

Title Associated Legendre Function of Second Kind $Q_n^{\ell}(x)$

Purpose Evaluation of Legendre Function of second kind and degree

Method The value is computed by means of the recurrence relation

$$Q_0(x) = \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right)$$

$$Q_1(x) = Q_0(x) \cdot x - 1$$

$$n\, Q_n(x) = (2n-1) \cdot x \cdot Q_{n-1}(x) - (n-1) Q_{n-2}(x)$$

Calling Statement PRINT FNA (N, X)

Parameter

N Degree

X Argument

Global variables None

Listing

```
LIST
FILE    *SLLEG2

0010 DEF FNA(N,X)K,Q0,Q1,Q
0020 LET Q=Q0=LOG(ABS((1+X)/(1-X)))/2
0030 LET Q1=Q0*X-1
0040 FOR K=2 TO N STEP 1
0050 LET Q=((K+K-1)*X*Q1-(K-1)*Q0)/K
0060 LET Q0=Q1
0070 LET Q1=Q
0080 NEXT K
0090 IF N=1 THEN 120
0100 LET FN*=Q
0110 GOTO 130
0120 LET FN*=Q1
0130 FNEND

END OF LISTING
```

Example of use

```
OLD +LEG2
LIST
FILE    +LEG2

0010 !             ####        ####.##    ######.############
0020 PRINT
0030 PRINT TAB(20);"LEGENDRE FUNCTION OF SECOND KIND"
0040 PRINT
0050 PRINT ,"DEGREE N","X","Q[N](X)"
0060 PRINT
0070 DISP "INPUT N,X";
0080 INPUT N,X
0090 PRINT USING 10,N,X,FNA(N,X)
0100 GOTO 70

END OF LISTING


LINK *SLLEG2,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

                    LEGENDRE FUNCTION OF SECOND KIND

                 DEGREE N        X                Q[N](X)

                0            0.00          0.000000000000
                0            0.10          0.100335347731
                0            0.50          0.549306144334
                0            0.75          0.972955074528
                1            0.00         -1.000000000000
                1            0.25         -0.936146797030
```

*SLCHYF

Title	Confluent Hypergeometric Function
Purpose	Evaluation of the confluent Hypergeometric Function
method	The value is computed by means of the representation

$$M(a,b,c) = \sum_{0}^{\infty} \frac{(a)_n}{(b)_n} \cdot \frac{x^n}{n!}$$

$$(y)_n = \frac{\Gamma(y+n)}{\Gamma(y)} \quad , \qquad y = a, b$$

Calling Statement	PRINT FNA (A, B, X)
Parameter	See Method
Global Variables	None
Listing	

```
OLD *SLCHYF
LIST
FILE    *SLCHYF

0010 DEF FNA(A,B,X)S,Q,K
0020 LET S=Q=1
0030 FOR K=1 TO 1000 STEP 1
0040 LET Q=Q*X*(A+K-1)/((B+K-1)*K)
0050 IF ABS(Q)<1E-14 THEN 80
0060 LET S=S+Q
0070 NEXT K
0080 LET FN*=S
0090 FNEND

END OF LISTING
```

Example of use

```
OLD +CHYF
LIST
FILE    +CHYF

0010 PRINT TAB(15);"CONFLUENT HYPERGEOMETRIC FUNCTION"
0020 PRINT
0030 PRINT ,"ARGUMENTS",,"VALUE"
0040 PRINT
0050 PRINT "  A","  B","  X","FNF(A,B,X)"
0060 PRINT
0070 DISP "INPUT A,B,X";
0080 INPUT A,B,X
0090 PRINT A,B,X,FNF(A,B,X)
0100 GOTO 70

END OF LISTING


LINK *SLCHYF,F
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
             CONFLUENT HYPERGEOMETRIC FUNCTION

                ARGUMENTS                          VALUE

  A               B               X              FNF(A,B,X)

 17              16              1               2.8881744
-1.3             .2              .1              .35821233
-.3              1.2             .1              .97459511
 .9              1.8             1               1.7232892
```

*SLGHYP

Title Gauss Hypergeometric Function

Purpose Evaluation of the Gauss Hypergeometric Series

Method The value is computed by means of the representation

$$F(a,b;c;x) = \sum \frac{(a)_n \cdot (b)_n}{(c)_n} \cdot \frac{x^n}{n!}$$

$$(y)_n = \frac{\Gamma(y+n)}{\Gamma(y)} \quad ; y = a, b, c$$

Calling Statement PRINT FNA (A, B, C, X)

Parameter See Method

Global variables None

Listing

```
OLD *SLGHYP
LIST
FILE    *SLGHYP

0010 DEF FNA(A,B,C,X)S,Q,K
0020 LET S=Q=1
0030 FOR K=1 TO 1000 STEP 1
0040 LET Q=Q*X*(A+K-1)*(B+K-1)/(K*(C+K-1))
0050 IF ABS(Q)<1E-14 THEN 80
0060 LET S=S+Q
0070 NEXT K
0080 LET FN*=S
0090 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE    +GHYP


0010 PRINT
0020 PRINT TAB(25);"GAUSS HYPERGEOMETRIC SERIES"
0030 PRINT
0040 PRINT TAB(30);"ARGUMENTS",,"VALUE "
0050 PRINT
0060 DISP "ENTER A,B,C,X";
0070 INPUT A,B,C,X
0080 PRINT A,B,C,X,FNG(A,B,C,X)
0090 GOTO 60

END OF LISTING


LINK *SLGHYP,G
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****
```

GAUSS HYPERGEOMETRIC SERIES

ARGUMENTS				VALUE
1.	1	2	.5	1.3862944
.5	1	1.5	.25	1.0986123
.5	1	1.5	-.25	.92723522
2	3	3	.5	4.0000000
2	2.5	.5	.36	19.607544
3	3.5	1.5	.81	11111.107
-5	5	.5	-81	1.8409361E+12
2	-1	.5	-.36	2.44

Title Sine Integral Si (x)

Purpose To calculate the function

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

Method An approximation serie is used

$$Si(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}$$

Calling Statement F = FNZ (X)

Parameters X = Value for which function will be calculated

Global variables None

Return The value of the function

Listing

```
OLD *SLSIF
LIST
FILE    *SLSIF


0010 DEF FNA(X)K,S,Q
0020 LET Q=S=X
0030 FOR K=1 TO 1000 STEP 1
0040 LET Q=-Q*X*X*(K+K-1)/((K+K)*(K+K+1)*(K+K+1))
0050 IF ABS(Q)<1E-15 THEN 80
0060 LET S=S+Q
0070 NEXT K
0080 LET FN*=S
0090 FNEND

END OF LISTING
```

Example of use

```
OLD +SIF
LIST
FILE    +SIF

0010 PRINT
0020 PRINT TAB(15);"SINE INTEGRAL SI"
0030 PRINT
0040 DISP "INPUT X";
0050 INPUT X
0060 PRINT ,"SI(";X;")=",FNA(X)
0070 PRINT
0080 GOTO 40

END OF LISTING


LINK *SLSIF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

              SINE INTEGRAL SI

                SI( 0 )=          0

                SI( .1 )=         9.3944461E-02

                SI( .2 )=         .19955609

                SI( .5 )=         .49310742

                SI( .6 )=         .58812881

                SI( .75 )=        .72695425

                SI( 1 )=          .94608307

                SI( 1.5 )=        1.3246835

                SI( 2 )=          1.6054130
```

*SLCINF

Title Cosine Integral Cin (x)

Purpose Calculate

$$Cin(x) = \int_0^x \frac{\cos t}{t}\, dt \qquad (|\arg x| < \pi)$$

Method The routine uses an approximation serie

$$cin(x) = \gamma + \ln x + \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{2n\,(2n)!}$$

γ is the Enter Constant

Calling Statement F = FNZ (X)

Parameter X = Upper limit of integration

Global variables None

Return The value of the function

Listing

```
OLD *SLCINF
LIST
FILE    *SLCINF


0010 DEF FNA(X)K,S,Q
0020 LET S=Q=-X*X/4
0040 FOR K=2 TO 1000 STEP 1
0050 LET Q=-Q*X*X*(K+K-2)/((K+K-1)*(K+K)*(K+K))
0060 IF ABS(Q)<1E-15 THEN 90
0070 LET S=S+Q
0080 NEXT K
0090 LET S=S+LOG(X)+5.772156649015E-1
0100 LET FN*=S
0110 FNEND

END OF LISTING
```

Example of use

```
OLD *CINF
LIST
FILE    *CINF


0010 PRINT
0020 PRINT TAB(15);"COSINE INTEGRAL CI"
0030 PRINT
0040 DISP "INPUT X";
0050 INPUT X
0060 PRINT ,"CI(";X;")=",FNA(X)
0070 PRINT
0080 GOTO 40

END OF LISTING


LINK *SLCINF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

              COSINE INTEGRAL CI

                CI( 0 )=         -9.9999999E+99

                CI( .001 )=      -6.3305399

                CI( 1.0000000E-13 )=             -29.356391



              COSINE INTEGRAL CI

                CI( .000001 )=   -13.238295

                CI( .1 )=        -1.7278684

                CI( .5 )=        -.17778408

                CI( .75 )=        .15216360

                CI( 1 )=          .33740392

                CI( 1.25 )=       .43430072

                CI( 1.5 )=        .47035632

                CI( 2 )=          .42298083

                CI( 2.5 )=        .28587120

                CI( 5 )=         -.19002975

                CI( 10 )=        -4.5456433E-02
```

*SLEIF

Title Exponential Integral Ei (x)

Purpose To calculate

$$Ei(x) = \int_{-x}^{\infty} \frac{e^{-t}}{t}\, dt \qquad x > 0$$

Method

$$Ei(x) = \gamma + \ln x + \sum_{n=1}^{\infty} \frac{x^n}{n\,n!}$$

γ = Enter Constant = 0.5772156649...

Calling Statement F = FNZ (X)

Parameter X = Inferior limit of integration

Global variables None

Return The value of the function

Listing

```
OLD *SLEIF
LIST
FILE    *SLEIF

0010 DEFFNA(X)E,Q,K
0020 LET E=Q=X
0030 FOR K=2 TO 1000 STEP 1
0040 LET Q=Q*X*(K-1)/(K*K)
0050 IF ABS(Q)<1E-15 THEN 80
0060 LET E=E+Q
0070 NEXT K
0080 LET E=E+0.5772156649+LOG(X)
0090 LET FN*=E
0100 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     TEST


0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LINK *SLEIF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X              F(X)
 .1             -1.6228128
 .5              .45421990
 1               1.8951178
 1.7             3.9209632
 2               4.9542344
 5               40.185275
 7               191.50474
```

*SLEINF

Title Exponential Integral Ein (x)

Purpose To calculate the value of

$$Ein(x) = -\int_x^\infty \frac{e^{-t}}{t}\,dt + \ln x + \gamma$$

where

γ = Enter Constant

Method

$$Ein(x) = -\sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n \cdot n!}$$

Calling Statement F = FNZ (x)

Parameter X = inferior limit of integration

Global variables None

Return The value of the function

```
OLD *SLEINF
LIST
FILE    *SLEINF

0010 DEF FNA(X)E,Q,K
0020 LET E=Q=-X
0030 FOR K=2 TO 1000 STEP 1
0040 LET Q=-Q*X*(K-1)/(K*K)
0050 IF ABS(Q)<1E-15 THEN 80
0060 LET E=E+Q
0070 NEXT K
0080 LET FN*=E
0090 FNEND

END OF LISTING
```

Example of use

```
OLD TEST
LIST
FILE     TEST


0005 PRINT
0010 PRINT " X","F(X)"
0020 DISP "ENTER X";
0030 INPUT X
0040 PRINT X,FNA(X)
0050 GOTO 20

END OF LISTING


LINK *SLEINF,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

 X               F(X)
 .1              -9.7554530E-02
 .2              -.19042830
 1               -.79659360
 1.5             -1.0827004
 2               -1.3192634
 3               -1.6888763
```

Title — Finding zeros of a real polynomial using Newton-Bairstow algorithm

Purpose — The routine calculates real or complex zeros of a polynomial with real coefficients

Method — A quadratic factor of the polynomial is determined, then the zeros of this factor are computed.

Let $f(x)$ be the polynomial, $g_k(x)$ an approximation to a quadratic factor of $f(x)$:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$$

$$g_k(x) = x^2 - p_k x - q_k$$

$g_{k+1}(x)$ is found by

$$p_{k+1} = p_k + \Delta p \qquad q_{k+1} = q_k + \Delta q$$

with

$$\Delta p = \frac{b_0 c_3 - b_1 c_2}{c_2^2 - c_3(c_1 - b_1)} \qquad \Delta q = \frac{b_1(c_1 - b_1) - b_0 c_2}{c_2^2 - c_3(c_1 - b_1)}$$

$$b_k = b_{k+1} p_k + b_{k+2} q_k + a_k$$

$$0 \le K \le n$$

$$c_k = c_{k+1} p_k + c_{k+1} q_k + b_k$$

$$c_{n+1} = c_{n+2} = b_{n+1} = b_{n+2} = 0$$

The algorithm stops when

i) $|\Delta p|$ and $|\Delta q| < E$

E: user-defined tolerance

$x^2 - p_{k+1} x - q_{k+1}$ is said to be the quadratic factor

or

ii) more than Z iterations are made without finding a proper factor

Z: user-defined max. number of iterations

Calling Statement — Γ = PNA (X,Y,E;Z,R)

Parameters

X : first approximation for P_k

Y : first approximation for Q_k

E : tolerance

Z : max. Number of iterations

R : deflation: 1 (YES), Ø (NO) (see 'option')

Global variables

Input	D		Degree
	E ( )		Vector of coefficients
Output	i)	Roots	R1 roots of the polynomial if they real
			R2 roots of the polynomial if they real
		or	
			R1 real part
			R2 ABS (imaginary part)
			if the roots are complex
	ii)	Value	P1 Real and imaginary part of the value
			P2 of the polynomial, computed for the roots found
	iii)	Factor	F1
			F2 the last computed values for p_k,qk

Status values

FN = -1 no solution, the polynomial is degenerate

FN = Ø no solution possible with the defined starting values

FN = 1 only one root: R1, R2 is Ø

FN = 2 two real roots

FN = 3 two complex roots

Option:

If R = 1, then all roots are computed, the original polynomial will be destroyed during computation.
If you need it for other computation you should shave it in a data file or in an another vector.

Listing

```
LIST
FILE    *SLBAIR

0002 DEF FNA(X,Y,E,Z,R)F,G,G1,G2,I,J,T,T0,T1
0004 IF D>0 THEN 10
0006 LET FN*=-1
0008 GOTO 152
0010 IF D>1 THEN 20
0012 LET R2=0
0014 LET R1=-E(1)/E(2)
0016 LET FN*=1
0018 GOTO 152
0020 IF D>2 THEN 28
0022 LET X=-E(2)/E(3)
0024 LET Y=-E(1)/E(3)
0026 GOTO 80
0028 FOR I=1 TO Z STEP 1
0030 LET F=Q1=G=G1=G2=0
```

```
0032 FOR J=D+1 TO 1 STEP -1
0034 LET Q2=Q1
0036 LET Q1=F
0038 LET F=Q1+X+Q2+Y+E(J)
0040 IF J=1 THEN 50
0042 LET G=Q1+X+Q2+Y+F
0044 IF J=2 THEN 50
0046 LET G2=G1
0048 LET G1=G
0050 NEXT J
0052 LET T=G1+G1-G2+(G-Q1)
0054 IF T<>0 THEN 62
0056 LET T0=.01+X+E
0058 LET T1=.01+Y+E
0060 GOTO 66
0062 LET T0=(F+G2-Q1+G1)/T
0064 LET T1=(Q1+(G-Q1)-F+G1)/T
0066 LET X=X+T0
0068 LET Y=Y+T1
0070 IF ABS(T0)>E THEN 74
0072 IF ABS(T1)<E THEN 80
0074 NEXT I
0076 LET FN+=0
0078 GOTO 148
0080 LET Q1=X+X+4+Y
0082 IF Q1<0 THEN 98
0084 LET FN+=2
0086 LET J=0
0088 LET R1=(X+SQR(Q1))/2
0090 LET R2=(X-SQR(Q1))/2
0092 GOSUB 130
0094 IF D=2 THEN 148
0096 GOTO 108
0098 LET FN+=3
0100 LET R1=X/2
0102 LET J=R2=SQR(ABS(Q1))/2
0104 GOSUB 130
0106 IF D=2 THEN 148
0108 IF R=0 THEN 148
0110 LET E(D+3)=E(D+2)=0
0112 FOR I=D+1 TO 3 STEP -1
0114 LET E(I)=E(I+1)*X+E(I+2)*Y+E(I)
0116 NEXT I
0118 FOR I=D-1 TO 1 STEP -1
0120 LET E(I)=E(I+2)
0122 NEXT I
0124 LET D=D-2
0126 GOTO 148
0130 LET P1=G1=E(D+1)
0132 LET P2=G2=0
0134 FOR I=D TO 1 STEP -1
0136 LET P1=G1*R1-G2*J+E(I)
0138 LET P2=G1*J+G2*R1
0140 LET G1=P1
0142 LET G2=P2
0144 NEXT I
0146 RETURN
0148 LET Q1=X
0150 LET Q2=Y
0152 FNEND

END OF LISTING
```

Example of use

```
OLD +BAIR
LIST
FILE    +BAIR


0010 REM  NEWTON-BAIRSTOW ALGORITHM
0020 PRINT
0030 DISP "ENTER DEGREE";
0040 INPUT D
0050 PRINT "DEGREE OF POLYNOMIAL =";D
0060 PRINT
0070 PRINT "COEFFICIENTS, IN DESCENDING ORDER"
0080 FOR I=D+1 TO 1 STEP -1
0090 DISP "ENTER C";I-1;
0100 INPUT E(I)
0110 PRINT "C";I-1,"=";E(I)
0120 NEXT I
0130 PRINT
0140 DISP "TOLERANCE";
0150 INPUT E
0160 DISP "NUMBER OF ITERATIONS";
0170 INPUT Z
0180 DISP "DEFLATION";
0190 INPUT R
0200 PRINT "TOLERANCE =";E;" ITERATIONS =";Z
0210 IF R=0 THEN 230
0220 PRINT "DEFLATION REQUESTED"
0230 PRINT
0240 FOR I=1 TO INT((D+1)/2) STEP 1
0250 DISP "APPROXIMATION P,Q";
0260 INPUT X,Y
0270 LET T=FNA(X,Y,E,Z,R)
0280 ON T+1 GOTO 310,370,390,430
0290 PRINT "DEGENERATE POLYNOMIAL"
0300 GOTO 9999
0310 PRINT "NO SOLUTION YET"
0320 DISP "ANOTHER TEST";
0330 INPUT X
0340 IF X=0 THEN 9999
0350 GOTO 140
0360 REM
0370 PRINT "ONE REAL ZERO:",R1
0380 GOTO 9999
0390 PRINT "TWO REAL ZEROS:",R1,R2
0400 PRINT "FACTORS:",Q1,Q2
0410 PRINT "VALUE:",P1,P2
0420 GOTO 490
0430 PRINT "TWO COMPLEX ZEROES:"
0440 PRINT ,R1,R2;"i"
0450 PRINT ,R1,-R2;"i"
0460 PRINT "FACTORS:",Q1,Q2
0470 PRINT "VALUE:",P1,P2;"i"
0480 IF R=0 THEN 9999
0490 PRINT
0500 NEXT I

END OF LISTING
```

```
LINK *SLBAIR,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

DEGREE OF POLYNOMIAL = 4

COEFFICIENTS, IN DESCENDING ORDER
C 4 = 1
C 3 = 0
C 2 = 0
C 1 = 0
C 0 =-1

TOLERANCE = .0001  ITERATIONS = 100
DEFLATION REQUESTED

TWO REAL ZEROS:  1               -1
FACTORS:         0                1
VALUE:           0                0

TWO COMPLEX ZEROES:
                 0                1 i
                 0               -1 i
FACTORS:         0               -1
VALUE:           0                0 i
```

Title

Zeros of scalar function using the bisection algorithm

Purpose

The routine calculates a root of scalar functions by bisectioning a user-defined interval. The function must be defined by DEF FNA (X) (multi - or single line).

Method

Let f (x) be the function, [a, b] the starting interval and E the tolerance.
The real number x is said to be a root, if one of the following conditions is fulfilled.

1) $f(x) = \emptyset$

or

2) i) $x = (a' + b') / Z$,
   ii) $[a',b'] \subset [a,b]$
   iii) $f(a') \cdot f(b') < \emptyset$
   iv) $|a' - b'| < E$

The interval [a', b'] is found by continued bisectioning of the original interval [a, b].
Let $c = (a + b) / Z$; [a, c] a [c, b]
will by taken as the new interval:

i) $f(a) \cdot f(c) > \emptyset$: a := c
   b := b

ii) $f(a) \cdot f(c) < \emptyset$: a := a
   b := c

If $f(a) \cdot f(c) = \emptyset$, a root is found.

Calling Statement

F = FNB (Q1,Q2,E)

Parameters

Q1, Q2 : Interval - limits
E : Tolerance

Return Values

FNB = $\emptyset$: a root is found
U = value
FNB = 1 : a pole is found
U = 9.99999 E 63 (dummy value)
FNB = 2 : no or a even number of roots
U = -9.99999 E 63 (dummy value)

Listing

```
FILE    *SLRBIS

0002 DEF FNB(Q1,Q2,E)Q3
0004 IF FNA(Q2)<>0 THEN 12
0006 LET FN=+0
0008 LET U=Q1
0010 GOTO 56
0012 LET Q3=(Q1+Q2)/2
0014 DISP "INTERVAL LENGTH";ABS(Q2-Q1)
0016 IF ABS(FNA(Q3))<(1*(E+E) THEN 24
0018 LET U=9.99999E63
0020 LET FN=+1
0022 GOTO 56
0024 ON SGN(FNA(Q1)*FNA(Q3))+2 GOTO 40,26,36
0026 IF FNA(Q1)<>0 THEN 32
0028 LET U=Q1
0030 GOTO 54
0032 LET U=Q3
0034 GOTO 54
0036 LET Q1=Q3
0038 GOTO 42
0040 LET Q2=Q3
0042 IF ABS(Q2-Q1)>E THEN 12
0044 IF FNA(Q1)*FNA(Q2)<=0 THEN 52
0046 LET FN=+2
0048 LET U=-9.99999E63
0050 GOTO 56
0052 LET U=(Q1+Q2)/2
0054 LET FN=+0
0056 FNEND

END OF LISTING
```

Example of use

```
FILE    *BISE1

0010 REM DEMONSTRATION FOR THE BISECTION ALGORITHM
0020 DISP "ENTER LIMITS";
0030 INPUT Q1,Q2
0040 PRINT
0050 PRINT "LIMITS:";Q1;Q2
0060 PRINT
0070 LET T=FNB(Q1,Q2,1E-5)
0080 PRINT "SOLUTION:";U
0090 DEF FNA(X)=X↑2-EXP(X)+8

END OF LISTING


LINK *SLRBIS/B
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

LIMITS:          0               100

SOLUTION:        2.7428551
```

Title	Roots of a complex polynomial
Purpose	To find one or all zeros of a polynomial with complex coefficients
Method	Let p(x) be the polynomial. The method is a quadratic Lagrangian interpalation: let x_{n-2}, x_{n-1}, x_n be three distinct points, i.e. three approximations of s, s zero of p (x). A quadratic is drawn through these points, the point closest to zero is taken as x_{n+1} Stopping criteria: i) $\lvert x_{n+1} - x_n \rvert < E$, E : user-defined tolerance ii) $p(x_n) = \emptyset$, a zero is found : x_n iii) the coefficients of the quadratic are zero : the quadratic is degenerated and has no zero iv) the number of iterations is larger than T, T : number of iterations, defined by the user.
Calling Statement	F = FNA (XØ, YØ, I1, E, Z)
Parameter	XØ : approximation for the real part YØ : approximation for the imaginary part I1 : maximal number of iterations E : user-defined tolerance Z : Z = 1 all zeros are calculated with one starting approximation Z = Ø only one zero is computed
Global variables input	R1 degree of the polynomial E() real part of the coefficients F() imaginary part of the coefficients

output	P()	real parts of the roots
	Q()	imaginary parts of the roots
	Z9	number of roots
calculation	G()	
	G()	may not be used outside the function
Status value	FN = -1	degenerated
	FN = 0	zero(s) are found
	FN = 1	no solution with II iterations and tolerance E
	FN = 2	coefficients of the quadratic are zero.

Listing

```
OLD *SLMULL
LIS
FILE    *SLMULL

0002 DEF FNB(A,B,C,D,E,I,J,T,X,Y,Q,N,U1,V1
0004 LET Q=1
0006 LET Z9=0
0008 DIM Q(2)
0010 IF P1>0 THEN 16
0012 LET FN=-1
0014 GOTO 368
0016 FOR I=1 TO P1+1 STEP 1
0018 IF ABS(E(I))+ABS(F(I))>0 THEN 26
0020 NEXT I
0022 LET FN=-1
0024 GOTO 368
0026 IF I=1 THEN 38
0028 LET I=I-1
0030 FOR J=Z9+1 TO Z9+I STEP 1
0032 LET P(J)=Q(J)=0
0034 NEXT J
0036 LET Z9=Z9+I
0038 IF I>0 THEN 48
0040 LET FN=0
0042 GOTO 368
0044 FEN
0046 FOR J=1 TO P1+1-I STEP 1
0048 LET E(J)=E(J+I)
0050 LET F(J)=F(J+I)
0052 NEXT J
0054 LET P1=P1-I
0056 IF P1>1 THEN 72
0058 IF P1=1 THEN 72
0060 LET S=E(2)*E(2)+F(2)*F(2)
0062 LET Z9=Z9+1
0064 LET P(Z9)=-(E(2)*E(1)+F(2)*F(1))/S
0066 LET Q(Z9)=-(E(2)*F(1)-E(1)*F(2))/S
0068 LET FN=0
0070 GOTO 316
0072 LET S=0
0074 FOR J=1 TO P1+1 STEP 2
0076 LET S=S+E(J)*E(J)+F(J)*F(J)
0078 NEXT I
```

```
0080 IF E>0 THEN 120
0082 LET E(1)=-E(1)
0084 LET F(1)=-F(1)
0086 LET G(17)=SGN(F(1))*(PI/2-1E-10)
0088 IF E(1)=0 THEN 92
0090 LET G(17)=ATN(F(1)/E(1))+(1-SGN(E(1)))*(PI/2-1E-10)
0092 LET G(20)=SQR(E(1)*E(1)+F(1)*F(1))
0094 LET G(18)=SGN(F(R1+1))*(PI/2-1E-10)
0096 IF E(R1+1)=0 THEN 100
0098 LET G(18)=ATN(F(R1+1)/E(R1+1))+(1-SGN(E(R1+1)))*(PI/2-1E-10)
0100 LET G(21)=SQR(E(R1+1)*E(R1+1)+F(R1+1)*F(R1+1))
0102 LET G(19)=EXP(LOG(G(20)/G(21))/R1)
0104 LET Z9=Z9+1
0106 FOR K=0 TO R1-1 STEP 1
0108 LET T=(G(17)-G(18)+2*PI*K)/R1
0110 LET P(Z9+K)=G(19)*COS(T)
0112 LET Q(Z9+K)=G(19)*SIN(T)
0114 NEXT K
0116 LET FN1=0
0118 GOTO 316
0120 REM MULLER'S METHOD
0122 LET Q=0
0124 LET G(37)=X0
0126 LET G(41)=Y0
0128 LET X=G(37)
0130 LET Y=G(41)
0132 GOSUB 348
0134 LET G(29)=U
0136 LET G(33)=V
0138 LET G(39)=G(37)
0140 LET G(43)=G(41)
0142 LET G(38)=G(37)+.1
0144 LET G(42)=G(41)
0146 LET X=G(38)
0148 LET Y=G(42)
0150 GOSUB 348
0152 LET G(30)=U
0154 LET G(34)=V
0156 LET G(11)=G(12)=0
0158 IF ABS(G(29))+ABS(G(33))=0 THEN 300
0160 LET X=G(39)=(G(37)+G(38))/2
0162 LET Y=G(43)=(G(41)+G(42))/2
0164 FOR Q=1 TO I1 STEP 1
0166 DISP "ROOT",Z9,"ITER#";Q
0168 GOSUB 348
0170 LET G(31)=U
0172 LET G(35)=V
0174 IF ABS(U)+ABS(V)=0 THEN 300
0176 LET A=(G(38)-G(37))*(G(38)-G(37))+(G(42)-G(41))*(G(42)-G(41))
0178 LET G(19)=((G(39)-G(38))*(G(38)-G(37))+(G(43)-G(42))*(G(42)-G(41)))/A
0180 LET G(20)=((G(43)-G(42))*(G(38)-G(37))-(G(39)-G(38))*(G(42)-G(41)))/A
0182 LET G(1)=G(19)*(G(30)-G(29))-G(20)*(G(34)-G(33))
0184 LET G(2)=G(20)*(G(30)-G(29))+G(19)*(G(34)-G(33))
0186 LET G(14)=G(31)-G(30)-G(1)
0188 LET N1=G(35)-G(34)-G(2)
0190 LET G(1)=G(19)*G(14)-G(20)*N1
0192 LET G(2)=G(20)*G(14)+G(19)*N1
0194 LET G(14)=G(19)*G(19)-G(20)*G(20)
0196 LET N1=G(19)+G(19)+1
0198 LET G(3)=N1*(G(31)-G(30))-2*G(20)*(G(35)-G(34))
0200 LET G(4)=N1*(G(35)-G(34))+2*G(20)*(G(31)-G(30))
0202 LET G(3)=G(3)+G(14)*(G(29)-G(30))-G(20)*(N1-1)*(G(33)-G(34))
0204 LET G(4)=G(4)+G(20)*(N1-1)*(G(29)-G(30))+G(14)*(G(33)-G(34))
0206 LET G(5)=(G(19)+1)*G(31)-G(20)*G(35)
0208 LET G(6)=(G(19)+1)*G(35)+G(20)*G(31)
0210 IF ABS(G(1))+ABS(G(2))<>0 THEN 222
0212 IF ABS(G(3))+ABS(G(4))=0 THEN 312
0214 LET N=G(3)*G(3)+G(4)*G(4)
0216 LET G(19)=-(G(5)*G(3)+G(6)*G(4))/N
0218 LET G(20)=(G(4)*G(5)-G(6)*G(3))/N
0220 GOTO 258
```

```
0222 LET G(0)=G(39)-G(38)
0224 LET G(10)=G(4)*G(2)
0226 LET G(7)=G(3)*G(3)-G(4)*G(4)+G(1)*G(1)-G(2)*G(5)
0228 LET G(8)=G(6)*G(2)+G(4)*G(3)-G(1)*G(5)
0230 LET R=SQR(G(7)*G(7)+G(8)*G(8))
0232 LET G(12)=SGN(G(8))*(PI/2-ATN(G(7)/G(8)))
0234 IF G(7)=0 THEN 238
0236 LET G(16)=ATN(G(8)/G(7))+(1-SGN(G(7)))*(PI/2+1E-10)
0237 LET G(25)=2*G(3)*G(16)/2
0238 LET G(26)=2*SIN(G(16)/2)
0240 LET N=2*G(1)*G(1)+G(2)*G(2)
0242 LET G(19)=(G(25)*G(3)+G(2)*G(1)*(G(26)-G(4)))/N
0244 LET G(20)=(G(2)*G(25)*G(3)+G(1)*(2*G(26)-G(4)))/N
0246 LET G(5)=-(G(3)*G(25)*G(1)-(2*G(26)-G(4))*G(2))/N
0250 LET G(6)=(G(3)*G(25)*G(2)-(2*G(26)-G(4))*G(1))/N
0251 IF G(19)*G(19)+G(20)*G(20)<=G(5)*G(5)+G(6)*G(6) THEN 258
0254 LET G(19)=G(5)
0256 LET G(20)=G(6)
0258 LET G(11)=G(9)*G(19)-G(10)*G(20)
0260 LET G(12)=G(8)*G(20)+G(10)*G(19)
0262 IF SQR(G(11)*G(11)+G(12)*G(12))<E THEN 300
0264 LET G(37)=G(38)
0266 LET G(39)=G(29)
0268 LET M=G(39)=G(39)+G(11)
0270 LET G(29)=G(38)
0272 LET G(38)=G(21)
0274 LET G(32)=G(34)
0276 LET G(24)=G(35)
0278 LET G(41)=G(42)
0280 LET G(42)=G(43)
0282 LET Y=G(43)=G(43)+G(12)
0284 NEXT Q
0286 LET Z=Q-1
0288 LET Z9=Z9+1
0290 LET P(Z9)=G(38)
0292 LET Q(Z9)=G(43)
0294 GOSUB 342
0296 LET FN=1
0298 GOTO 368
0300 LET Z9=Z9+1
0302 LET X=P(Z9)=G(38)+G(11)
0304 LET Y=Q(Z9)=G(43)+G(12)
0306 GOSUB 348
0308 LET FN=0
0310 GOTO 316
0312 LET FN=2
0314 GOTO 368
0316 IF Z9=0 THEN 328
0318 GOSUB 348
0320 GOTO 368
0322 IF R1=1 THEN 328
0324 GOSUB 348
0326 GOTO 368
0328 FOR I=R1 TO 1 STEP -1
0330 LET E(I)=E(I+1)*P(Z9)-Q(Z9)*F(I+1)+E(I)
0332 LET F(I)=F(I+1)*P(Z9)+E(I+1)*Q(Z9)+F(I)
0334 NEXT I
0336 FOR I=1 TO R1 STEP 1
0338 LET E(I)=E(I+1)
0340 LET F(I)=F(I+1)
0342 NEXT I
0344 LET R1=R1-1
0346 GOTO 16
0348 REM   EVALUATION
0350 LET U1=E(R1+1)
0352 LET V1=F(R1+1)
0354 FOR I=R1 TO 1 STEP -1
0356 LET U=U1*M-V1*Y+E(I)
0358 LET V=V1*M+U1*Y+F(I)
0360 LET U1=U
0362 LET V1=V
0364 NEXT I
0366 RETURN
0368 FNEND

END OF LISTING
```

Example of use

```
OLD *MULL
LIS
FILE     *MULL


0010 DISP "ENTER DEGREE";
0020 INPUT R
0030 PRINT "ENTER ROOTS: X+iY IS ENTERED AS X,Y"
0040 FOR I=1 TO R @STEP 1
0050 DISP "ENTER ";"P(";I;")";
0060 INPUT E(I),F(I)
0070 PRINT E(I),F(I)
0080 NEXT I
0090 GOSUB 220
0100 PRINT
0110 PRINT "COEFFICIENTS:"
0120 FOR I=1 TO R+1 STEP 1
0130 PRINT I-1,A(I),B(I)
0140 NEXT I
0150 PRINT
0160 PRINT
0170 FOR I=1 TO R+1 STEP 1
0180 LET E(I)=A(I)
0190 LET F(I)=B(I)
0200 NEXT I
0210 GOTO 420
0220 REM    COMPUTATION OF COEFFICIENTS OF A COMPLEX POLYNOMIAL FROM ROOTS
0230 FOR K=2 TO R+1 STEP 1
0240 LET A(K)=0
0250 LET B(K)=B(2)=0
0260 NEXT K
0270 LET A(1)=-E(1)
0280 LET B(1)=-F(1)
0290 LET A(2)=1
0300 FOR J=2 TO R STEP 1
0310 FOR K=J+1 TO 2 STEP -1
0320 LET U=A(K)
0330 LET V=B(K)
0340 LET A(K)=A(K-1)-U*E(J)+V*F(J)
0350 LET B(K)=B(K-1)-U*E(J)-U*F(J)
0360 NEXT K
0370 LET U=A(1)
0380 LET V=B(1)
0390 LET A(1)=-U*E(J)+V*F(J)
0400 LET B(1)=-U*F(J)-V*E(J)
0410 NEXT J
0420 RETURN
0430 LET Q9=R1=R
0440 LET L1=FNQ(0,0,100,1E-5,1)
0450 PRINT "ROOT","REAL PART","IMAG.PART"
0460 FOR I=1 TO Q8 STEP 1
0470 PRINT I,P(I),Q(I)
0480 NEXT I
0490 PRINT
0500 PRINT
0510 PRINT "VALUE:",U,V

END OF LISTING
```

Some remarks:

The program + MULL calculates first the coefficients of a complex polynomial from roots, then the roots of this polynomial are computed by the routine *SLMULL

```
LOAD *SLMULL.S
0000 END
RUN
**** FORMALLY CORRECT PROGRAM ****
ENTER ROOTS   +0, IS ENTERED AS 0,0
 1                1
 2                2
 1               -.5
 0                1

COEFFICIENTS
 0               -5.5                .5
 1                6                 -9.5
 2                .5                 11.5
 3               -2                 -4.5
 4                1                  0

ROOT             REAL PART          IMAG.PART
 1                2.3721315E-10      1.0000000
 2                1.0000000         -.50000000
 3                1.0000000          1.0000000
 4                2.0000000          0.0000000

VALUE             0                 -1.0000000E-12
```

*SLNLIN

Title

Solution of a non-linear System

Purpose

To calculate the solution of a system of non-linear functions. The functions have to be defined by the function FNZ (see remark of the end)

Method

The solution is found by Newton - Iteration:

Let $\bar{x}_o$ an approximation for the solution-vector $\bar{x}$. The i-th approximation $\bar{x}_i$ is calculated by:

(1) $\bar{x}_i = \bar{x}_{i-1} - A \cdot F(\bar{x}_{i-1})$, $i \geq 1$

with:

F : system of functions

A : $J^{-1}(\bar{x})$

J (x) : Jacobi-Matrix of the System F;

(1) is equivalent with

(2) $J(\bar{x}_{i-1}) \cdot (\bar{x}_i - \bar{x}_{i-1}) = F(\bar{x}_{i-1})$

Now the linear System (2) is solved for $(\bar{x}_i - \bar{x}_{i-1})$ by means of the Crout - Algorithm.

Statement

T = FNA (N,X,E)

Parameter

N = order of the system

X = number of iterations

E = tolerance

Global variable

Input

X ( ) Vector of approximations

F ( ) array of functions

Output

X ( ) solution vector

F ( ) vector of the values of the functions

Status value

FN = ø Solution

FN = 1 no solution with N iteration and E tolerance

FN = 2 Jacobi-Matrix is singular

FN = 3 Algorithm diverges

REMARK:

Let $f_1 (x_1,\ldots, x_N)$, $f_2 (x_1,\ldots, x_N)$, ...,
$f_N (x_1,\ldots, x_N)$ the functions:

The system has to be defined the following way:

```
8000  DEF  FNZ  (X)
8010  F (1)  =      exp (x(1),...,x(N))
8020  F (2)  =      exp (x(1),...,x(N))
      :
      :
8 ... F (N)  =      exp (x(1),..., x(N))
      FN*  =  ø
      FNEND
```

The values are calculated by the statement

```
T  =  FNZ  (ø)
```

When the functions are defined in this way, the functions can be used outside the routine too.

"exp" means Expression, not the exponential function. (see ex.)

Listing

```
CLS *SLNLIN
LIS
FILE     *SLNLIN

0001 DEF FNA(N,X,E,A,E1,M,N1,I,J,K,L,N,S,T
0002 DATA -2,1,3,1,-3,1,2,3,1,1,-1,1
0005 LET H=.001
0008 FOR Q=1 TO N STEP 1
0010 REM CALCULATION OF JACOBIAN******
0012 FOR I=1 TO N STEP 1
0014 FOR J=1 TO N STEP 1
0016 LET M(I,J)=2
0018 NEXT J
0020 NEXT I
0022 FOR J=1 TO N STEP 1
0024 FOR I=1 TO 4 STEP 1
0026 READ A
0028 LET X(J)=X(J)+A*H
0030 LET T=FNZ(0)
0032 READ S,T
0034 FOR K=1 TO N STEP 1
0036 LET K(K,J)=K(K,J)*T+F(K)*S
0038 NEXT K
0040 NEXT I
0042 RESTORE
0044 LET X(J)=X(J)-2*H
```

```
0046 NEXT J
0048 LET T=FNC(0)
0050 LET H1=12*H
0052 FOR I=1 TO N STEP 1
0054 FOR J=1 TO N STEP 1
0056 LET K(I,J)=K(I,J)/H1
0058 NEXT J
0060 LET F(I)=-F(I)
0062 LET P(I)=I
0064 NEXT I
0066 REM END OF JACOBIAN******
0068 REM SOLUTION GROUT METHOD******
0070 FOR I=1 TO N STEP 1
0072 IF I=1 THEN 90
0074 FOR J=I TO N STEP 1
0076 LET S=0
0078 FOR K=1 TO I-1 STEP 1
0080 LET S=S+K(P(J),K)*K(P(K),I)
0082 NEXT K
0084 LET K(P(J),I)=K(P(J),I)-S
0086 NEXT J
0088 IF I=N THEN 130
0090 LET E1=ABS(K(P(I),I))
0092 LET L=P(I)
0094 LET M=I
0096 FOR J=I+1 TO N STEP 1
0098 IF ABS(K(P(J),I))<=E1 THEN 106
0100 LET E1=ABS(K(P(J),I))
0102 LET L=P(J)
0104 LET M=J
0106 NEXT J
0108 IF E1<1E-20 THEN 164
0110 LET P(M)=P(I)
0112 LET P(I)=L
0114 FOR J=I+1 TO N STEP 1
0116 LET S=0
0118 IF I=1 THEN 126
0120 FOR K=1 TO I-1 STEP 1
0122 LET S=S+K(P(I),K)*K(P(K),J)
0124 NEXT K
0126 LET K(P(I),J)=(K(P(I),J)-S)/K(P(I),I)
0128 NEXT J
0130 IF ABS(K(P(N),N))<1E-20 THEN 164
0132 LET S=0
0134 IF I=1 THEN 142
0136 FOR K=1 TO I-1 STEP 1
0138 LET S=S+K(P(I),K)*F(P(K))
0140 NEXT K
0142 LET F(P(I))=(F(P(I))-S)/K(P(I),I)
0144 NEXT I
0146 FOR I=N-1 TO 1 STEP -1
0148 LET S=0
0150 FOR K=I+1 TO N STEP 1
0152 LET S=S+K(P(I),K)*F(P(K))
0154 NEXT K
0156 LET F(P(I))=F(P(I))-S
0158 NEXT I
0160 LET T=0
0162 GOTO 166
0164 LET T=1
0166 REM
0168 IF T<>1 THEN 174
0170 LET FN*=2
0172 GOTO 206
0174 LET T=0
0176 FOR I=1 TO N STEP 1
0178 LET X(I)=X(I)+F(P(I))
0180 LET T=T+F(I)*F(I)
0182 NEXT I
0184 DISP @;"ITER",T;"INC."
0186 IF T<E THEN 198
0188 IF T>1/(E*E) THEN 202
0190 NEXT @
```

```
0192 LET T=FNC(0)
0194 LET FN+=1
0196 GOTO 206
0198 LET FN+=2
0200 GOTO 206
0202 LET FN+=3
0204 GOTO 206
0206 FNEND

END OF LISTING
```

Example of use

```
OLD *NLIN
_!!
FILE    *NLIN


0010 REM    ***** SOLUTION OF NONLINEAR SYSTEMS *****
0020 DISP "ENTER ORDER OF THE SYSTEM";
0030 INPUT N
0040 DISP "ENTER MAX. NUMBER OF ITERATIONS";
0050 INPUT I
0060 DISP "ENTER TOLERANCE";
0070 INPUT E
0080 PRINT "ORDER OF THE SYSTEM";N
0090 PRINT "NUMBER OF ITERATIONS ";I
0100 PRINT "TOLERANCE";E
0110 PRINT
0120 PRINT "ENTER APPROXIMATION-VECTOR"
0130 FOR I=1 TO N STEP 1
0140 DISP "ENTER X(";I;")";
0150 INPUT X(I)
0160 PRINT "X(";I;")=";X(I)
```

```
0170 NEXT I
0180 PRINT
0190 ON FNA(N,X,0) GOTO 260,340,370
0200 PRINT
0210 PRINT TAB(6);"SOLUTION:"
0220 FOR I=1 TO N STEP 1
0230 PRINT "X(";I;")=";USING "-#.########^^^^";X(I)
0240 NEXT I
0250 GOTO 9999
0260 PRINT "NO SOLUTION WITH";Z;"ITERATIONS AND THE TOLERANCE";E
0270 PRINT
0280 PRINT "LAST VALUES:"
0290 LET T=FNZ(0)
0300 FOR I=1 TO N STEP 1
0310 PRINT I,X(I)
0320 NEXT I
0330 GOTO 9999
0340 PRINT "JACOBIAN MATRIX OF THE SYSTEM IS SINGULAR"
0350 PRINT
0360 GOTO 9999
0370 PRINT "SOLUTION NOT POSSIBLE  THE ALGORITHM DIVERGES"
0380 DEF FNZ(D)
0390 LET F(1)=X(1)+X(1)*X(2)+X(2)-4
0400 LET F(2)=X(1)*X(2)-1
0410 LET FN=0
0420 FNEND

END OF LISTING


LINK *SLNLIN,A
0000 END
RUN
**** FORMALLY CORRECT PROGRAM ****
INIT. IT THE SYSTEM              :
NUMBER OF ITERATIONS:            50
TOLERANCE                        .000001

ENTER APPROXIMATION-VECTOR:
X( 1 )=          0
X( 2 )=          1


       SOLUTION
X( 1 )= .5177E7999               F( 1 )=-2.31441055E-06
X( 2 )= 1.93138517               F( 2 )=-2.31440375E-06
```

Title Lagrangian polynomial curve fitting

Purpose To calculate, for a given set of n data points (x_i, y_i) $i = 1,2 \ldots n$, the coefficients of the polynomial of degree (n-1) that passes through the given points

Method Let $p(x) = a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_1 x + a_o$ (1)
the n data points are used to determine the coefficients a_o, $a_1 \ldots a_{n-1}$. The Lagrange interpolation polynomial p (x) which is appropriate for interpolation is the following:

$$P(x) = \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} \cdot y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} \cdot y_2 + \vdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})} \cdot y_n \qquad (2)$$

Each term in the right side of equation (2) can be expanded in a recursive manner into a polynomial of degree (n-1) in the same form as equation (1). If one defines:

$$A_i = \prod_{\substack{j=1 \\ j \neq i}}^{n} (x_i - x_j)$$

the i-th term on the right-hand side of equation (2) is:

$$\frac{\left[\prod_{\substack{j=1 \\ j \neq i}}^{n} (x - x_j)\right] \cdot y_i}{A_i}$$

Let

$$\prod_{\substack{j=1 \\ j \neq i}}^{n} (x - x_j) = b_{i,n-1} x^{n-1} + b_{i,n-2} x^{n-2} + \ldots + b_{i,1} x + b_{i0}$$

these bij can be computed in a recursive fashion by successively multiplying one or more term. In particular, suppose for simplicity $i > K + 1$ and $1 < K + 1 < n$, then:

$B_{n-1} = B_{n-2} = \ldots = B_{k+1} = 0$, $B_k \ldots, B_1, B_o$
are the coefficients of

$$B = \prod_{\substack{j=1 \\ j \neq i}}^{K} (x - x_j)$$

which is the product of the first K-factors of

$$\prod_{\substack{j=1 \\ j \neq i}}^{K} (x - x_j)$$

When B is multiplied by the factor ($x - x_{K+1}$), a new polynomial B with the following coefficients is obtained:

$$\begin{aligned}
B'_{n-1} &= 0 \\
B'_{n-2} &= 0 \\
&\vdots \\
B'_{K+1} &= B_K \\
B'_K &= B_{K-1} - x_{K+1} B_K \\
B'_{K-1} &= B_{K-2} - x_{K+1} B_{K-1} \\
&\vdots \\
B'_1 &= B_0 - x_{K+1} B_1 \\
B'_0 &= - x_{K+1} B_0
\end{aligned}$$

In this way, each term in equation (2) is expanded and the coefficients are gathered by:

$$a_j = \sum_{i=1}^{n} \frac{b_{i,j}}{A_i} y_i \qquad 0 \leq i \leq (n-1)$$

Calling Statement

F = FNA (P)

Parameters

P = Number of points

Global variables

-input

E ( ) = array of x-values (one dimension)

F ( ) = array of y-values (one dimension)

-return

G ( ) = array of the coefficients of interpolating polynomial (one dimension)

-work

H ( ) = array of the coefficients of the development of every part of the Lagrange's formula.

Status value Ø correct calculation
1 denominator = Ø

Listing

```
LIST
FILE    *SLPLYF

0002 DEF FNA(P)C,I,J,T0,K,S,A
0004 FOR I=1 TO P STEP 1
0006 LET G(I)=0
0008 NEXT I
0010 IF P=1 THEN 92
0012 LET C=0
0014 FOR I=1 TO P STEP 1
0016 FOR J=1 TO P STEP 1
0018 LET H(J)=0
0020 NEXT J
0022 IF I=1 THEN 28
0024 LET H(1)=-E(1)
0026 GOTO 32
0028 LET H(1)=-E(2)
0030 LET C=1
0032 LET H(2)=1
0034 FOR J=2 TO P STEP 1
0036 IF ABS(I-1)+ABS(J-2)=0 THEN 62
0038 IF I<>J THEN 44
0040 LET C=1
0042 GOTO 62
0044 LET T0=H(1)
0046 LET H(1)=-E(J)*H(1)
0048 FOR K=2 TO J STEP 1
0050 LET S=H(K)
0052 LET H(K)=T0-E(J)*H(K)
0054 LET T0=S
0056 NEXT K
0058 IF C=1 THEN 62
0060 LET H(J+1)=1
0062 NEXT J
0064 LET C=0
0066 LET A=1
0068 FOR J=1 TO P STEP 1
0070 IF I=J THEN 74
0072 LET A=A*(E(I)-E(J))
0074 NEXT J
0076 IF A<>0 THEN 82
0078 LET FN*=1
0080 GOTO 96
0082 FOR J=1 TO P STEP 1
0084 LET G(J)=H(J)*F(I)/A+G(J)
0086 NEXT J
0088 NEXT I
0090 GOTO 94
0092 LET G(1)=F(1)
0094 LET FN*=0
0096 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RRPLYF

0010 DCL S (P,I1)
0020 :      ###         ###.####       ###.####
0030 DIM E(30),F(30),G(30),H(30)
0040 FILES VVVV
0050 PRINT
0060 SETW :1 TO 5
0070 READ :1,P
0080 PRINT "NUMBER OF POINTS  =";P
0090 PRINT
0100 FOR I1=1 TO P STEP 1
0110 READ :1,X,Y,W
0120 LET E(I1)=X
0130 LET F(I1)=Y
0140 PRINT USING 20,I1,E(I1),F(I1)
0150 NEXT I1
0170 IF FNA(P)=0 THEN 200
0180 DISP "DENOMINATOR=0"
0190 GOTO 9999
0200 PRINT TAB(20),"COEFFICIENTS OF POLYNOMIAL (IN DESCENDING ORDER)"
0210 PRINT
0220 FOR I1=P TO 1 STEP -1
0230 PRINT I1-1,G(I1)
0240 NEXT I1
0250 GOTO 9999

END OF LISTING


LINK *SLPLYF,A
9999 END


RUN
**** FORMALLY CORRECT PROGRAM ****

NUMBER OF POINTS  = 8

       1           0.0000          0.0000
       2           1.0000          1.0000
       3           2.0000          4.0000
       4           3.0000          9.0000
       5           4.0000         16.0000
       6           5.0000         25.0000
       7           6.0000         36.0000
       8           7.0000         49.0000
                  COEFFICIENTS OF POLYNOMIAL (IN DESCENDING ORDER)

 7              -4.8000000E-14
 6               4.0000000E-13
 5              -2.0000000E-12
 4              -3.3000000E-11
 3              -2.0000000E-11
 2               1
 1               0
 0               0
```

Ao	Bo	N	X_1	Y_1	W_1	X_2	Y_2	W_2	- - - -

File VVVV

Ao = first x - value

Bo = last x - value

N = number of points on data file

X_i = x value of point i i = 1, 2 ... N

Y_i = y value of point i

W_i = weight value of point i

```
EXE FLPRINT, UUUU
 0                7                0                0                0
 1                1                1                1                2
 4                1                3                9                1
 4                16               1                5                25
 1                6                36               1                7
 49               1                0                0                0
 0                0                0                0                0
 0                0                0                0                0
 0                0                0                0                0
 0                0                0                0                0
READY
```

Title — Lagrangian Interpolation

Purpose — To interpolate a table of values with a fixed increment by using a kwon Lagrange interpolating polynomial

Method — Fixed the lower and upper limits of an interval $x_1 - x_n$ and the increment h, the routine calculates

$$p(x_1),\ P(x_1+h) \ldots\ p(x_1+ih) \qquad x_1+ih \le x_N < x_1+(i+1)h$$

where P (x) is a Lagrangian polynomial

$$p(x) = a_{N-1}\,x^{N-1} + a_{N-2}\,x^{N-2} + \ldots + a_1 x + a_0$$

fitted by using N data points (see *SLPLYF).
It is possible interpolate a single point if

$$x_1 = x_n$$
$$h = 1$$

Calling Statement — F = FNB (A1,B1,H1,N)

Parameters
- A1 lower limit of interval
- B1 upper limit of interval
- H1 increment
- N number of Lagrangian coefficients

Global variables
- input
  - G ( ) coefficients of Lagrangian polynomial
- returns
  - O ( ) array of interpolated points (one dimension)
  - P ( ) array of Lagrangian polynomial values (one dimension)
  - P1 number of points interpolated in the given interval

Status value — ∅

Note - The routine uses a FNC function

Listing

```
LIST
FILE    *SLLAGI

0002 DEF FNB(A1,B1,H1,N)X
0004 LET P1=0
0006 FOR X=A1 TO B1 STEP H1
0008 LET P1=P1+1
0010 LET Q(P1)=X
0012 LET P(P1)=FNC(X,N)
0014 NEXT X
0016 LET FN*=P1
0018 FNEND
0020 DEF FNC(X,N)I,J
0022 LET J=G(N)
0024 IF N-1=0 THEN 32
0026 FOR I=1 TO N-1 STEP 1
0028 LET J=J*X+G(N-I)
0030 NEXT I
0032 LET FN*=J
0034 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RRLAGI

0010 DCL S (P,I1)
0020 :      ###          ###.####       ###.####
0030 DIM E(30),F(30),G(30),H(30),Q(50),P(50)
0040 FILES 0000
0050 PRINT
0060 SETW :1 TO 1
0070 READ :1,A0,B0,P
0080 PRINT "NUMBER OF POINTS  =";P
0090 PRINT
0100 FOR I1=1 TO P STEP 1
0110 READ :1,X,Y,W
0120 LET E(I1)=X
0130 LET F(I1)=Y
0140 PRINT USING 20,I1,E(I1),F(I1)
0150 NEXT I1
0160 IF FNA(P)=0 THEN 190
0170 DISP "DENOMINATOR=0"
0180 GOTO 9999
0190 PRINT TAB(20),"COEFFICIENTS OF POLYNOMIAL (IN DESCENDING ORDER)"
0200 PRINT
0210 FOR I1=P TO 1 STEP -1
0220 PRINT I1-1,G(I1)
0230 NEXT I1
0240 PRINT
0250 PRINT TAB(20);"LAGRANGIAN INTERPOLATION"
0260 PRINT
0270 DISP "INT.CODE:1=POINT,2=TABLE,0=STOP";
0280 INPUT R
0290 ON R+1 GOTO 9999,300,390
0300 DISP "X";
0310 INPUT A1
```

```
0320 IF (A0-A1)*(A1-B0)>=0 THEN 360
0330 DISP "OUTSIDE THE INTERVAL";A0;"-";B0
0340 DELAY 20
0350 GOTO 300
0360 LET B1=A1
0370 LET H1=1
0380 GOTO 480
0390 DISP "INT.INTERVAL(X1,Xn)";
0400 INPUT A1,B1
0410 DISP "INCREMENT";
0420 INPUT H1
0430 IF A1<A0 THEN 450
0440 IF B1<=B0 THEN 480
0450 DISP "TABLE OUTSIDE THE INTERVAL";A0;"-";B0
0460 DELAY 20
0470 GOTO 390
0480 DISP "LAGRANGIAN INTERP. IS RUNNING"
0485 LET Y=FNB(A1,B1,H1,P)
0490 FOR I=1 TO P1 STEP 1
0500 PRINT O(I),P(I)
0510 NEXT I
0520 GOTO 270

END OF LISTING


LINK *SLPLYR,A
LINK *SLLAGI,B
9999 END



RUN
**** FORMALLY CORRECT PROGRAM ****

NUMBER OF POINTS  = 8

        1           0.0000          0.0000
        2           1.0000          1.0000
        3           2.0000          4.0000
        4           3.0000          9.0000
        5           4.0000         16.0000
        6           5.0000         25.0000
        7           6.0000         36.0000
        8           7.0000         49.0000
                 COEFFICIENTS OF POLYNOMIAL (IN DESCENDING ORDER)

 7              -4.8000000E-14
 6               4.0000000E-13
 5              -2.0000000E-12
 4              -3.3000000E-11
 3              -2.0000000E-11
 2               1
 1               0
 0               0
```

```
                    LAGRANGIAN INTERPOLATION

INT.CODE:1=POINT,2=TABLE,0=STOP?
1
X?
15
OUTSIDE THE INTERVAL 0 - 7
X?
2
LAGRANGIAN INTERP. IS RUNNING
 2                  4.0000000
INT.CODE:1=POINT,2=TABLE,0=STOP?
2
INT.INTERVAL(X1,Xn)?
0,9
INCREMENT?
1
TABLE OUTSIDE THE INTERVAL 0 - 7
INT.INTERVAL(X1,Xn)?
0,2
INCREMENT?
.25
LAGRANGIAN INTERP. IS RUNNING
 0                  0
 .25                6.2500000E-02
 .5                 .25000000
 .75                .56250000
 1                  1.0000000
 1.25               1.5625000
 1.5                2.2500000
 1.75               3.0625000
 2                  4.0000000
INT.CODE:1=POINT,2=TABLE,0=STOP?
0
```

*SLFTRP

Title Coefficients of Fourier Series to represent discrete data

Purpose To compute the coefficients of Fourier Series to represent discrete data equally spaced over the interval $(0, 2\pi)$

Method If a function f (x) is defined at a set of 2N equally spaced points:

$$x_0, x_0+h, x_0+2h, \ldots\ldots$$

the transformation

$$x' = \left(\frac{\pi}{N}\right) y \qquad \text{where} \qquad y = \left(\frac{x'-x_0}{h}\right)$$

gives unit spacing and an approximation to function is given by the sum of first M harmonics of Fourier expansion:

$$f_M(y) = \frac{a_0}{2} + \sum_{m=1}^{M} \left(a_m \cos\frac{\pi}{N} my + b_m \sin\frac{\pi}{N} my\right)$$

where:

$0 \leqslant M \leqslant N-1$ for even number of observations

and $0 \leqslant M \leqslant N$ for odd number of observations

The user enters the maximum number of harmonics to be computed and the tolerance used to control the least squares error.
The a's and b's coefficients are calculated by using the Goertzel Algorithm, what gives a recursive method of calculating.
It is necessary to deal with to cases:

a) Odd number of observation (2N+1)

Let:

$Co = 1$ $\qquad$ $So = 0$

$C1 = \cos\left(\frac{2\pi}{2N+1}\right)$ $\qquad$ $S1 = \sin\left(\frac{2\pi}{2N+1}\right)$

$C_{K+1} = C_1 C_K - S_1 S_K$ $\qquad$ $S_{K+1} = C_1 S_K + S_1 C_K$

Law of cosine $\qquad$ Law of sines

the routine calculates:

$U_{2N+2} = U_{2N+1} = 0$

$U_m = f(m) + 2C_K U_{m+1} - U_{m+2}$

$m = 2N, 2N-1, \ldots. 1$

then:

$$a_K = \frac{2}{2N+1}\left[f(0) + C_K U_1 - U_2\right]$$

$$b_K = \frac{2}{2N+1} S_K U_1 \qquad K = 0, 1, \ldots N$$

b) even number of observation (2N)

Let:

$$C_0 = 1 \qquad S_0 = 0$$

$$C_1 = C_0 \frac{\pi}{N} \qquad S_1 = \sin\frac{\pi}{N}$$

$$C_{K+1} = C_1 C_K - S_1 S_K \qquad S_{K+1} = C_1 S_K + S_1 C_K$$

Law of cosines Law of sines

the routine calculates

$$U_{2N} = U_{2N+1} = 0$$

$$U_m = f(m) + 2 C_K U_{m+1} - U_{m+2}$$

$$m = 2N-1, 2N-2, \ldots 1$$

then:

$$a_K = \frac{1}{N}\left[f(0) + C_K U_1 - U_2\right]$$

$$b_K = \frac{1}{N} S_K U_1 \qquad K = 0, 1, \ldots N$$

The least square error E_M is computed for each M, where $0 \leq M \leq N-1$ for even number of observations and $0 \leq M \leq N$ for odd number of observations. Using only terms up to M, the least square error or standard deviation is given by:

$$E_M = \sum_y f^2(y) - N\left[\frac{a_0^2}{2} + \sum_{m=1}^{N}\left(a_m^2 + b_m^2\right)\right]$$

By observing the behavior of E_M as M increases it is possible to estimate the necessity of taking additional terms in the Fourier expansion.

The coefficients are computed to order M or until the least squares error is less than the given tolerance

Calling Statement F = FNA (M,E,N)

Parameter	M	number of harmonics
	E	tolerance
	N	number of observations

Global variables

- input	F ( )	array of function values (one dimension)
- return	G ( )	coefficients a_K (one dimension)
	H ( )	coefficients b_K (one dimension)
	E ( )	least squares error at step 1, 2... K (one dimension)
	P1	number of harmonics computed

Function value	Ø	correct calculation
	1	coefficient not computed, because

$M > P$ ($P = \frac{N}{2}$ if N even, $P = \frac{N-1}{2}$ if N odd)

Listing

```
LIST
FILE    *SLFTRP

0002 DEF FNA(M,E,N)I,D,C,S,C1,S1,U3,U2,P,A9,F,Q
0004 LET FN*=0
0006 LET F=0
0008 FOR I=1 TO N STEP 1
0010 LET F=F+F(I)*F(I)
0012 NEXT I
0014 IF N-INT(N/2)*2=1 THEN 20
0016 LET P=N/2
0018 GOTO 22
0020 LET P=(N-1)/2
0022 IF M<=P THEN 28
0024 LET FN*=1
0026 GOTO 78
0028 LET C=1
0030 LET S=0
0032 LET C1=COS(2*PI/N)
0034 LET S1=SIN(2*PI/N)
0036 FOR P1=1 TO M+1 STEP 1
0038 LET U3=U2=0
0040 FOR I=N TO 2 STEP -1
0042 LET U1=F(I)+2*C*U2-U3
0044 LET U3=U2
0046 LET U2=U1
0048 NEXT I
0050 LET G(P1)=2/N*(F(1)+C*U2-U3)
0052 LET H(P1)=2/N*S*U2
0054 LET Q=C1*C-S1*S
0056 LET S=C1*S+S1*C
0058 LET C=Q
0060 LET A9=0
0062 IF P1=1 THEN 70
0064 FOR I=2 TO P1 STEP 1
```

```
0066 LET A9=A9+G(I)*G(I)+H(I)*H(I)
0068 NEXT I
0070 LET E(P1)=F-P*(G(1)*G(1)/2+A9)
0072 IF ABS(E(P1))<E THEN 78
0074 NEXT P1
0076P1=P1-1
0078 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RRFTRP


0010 FILES ZZZZ
0020 DIM G(21),H(21),E(21),F(21),X(21)
0030 SETW :1 TO 1
0040 READ :1,A,B,N
0050 LET D=(B-A)/(N-1)
0060 PRINT TAB(15);" DATA  POINTS VALUES"
0070 PRINT
0080 FOR I=1 TO N STEP 1
0090 READ :1,X(I),F(I),W
0100 PRINT I,X(I),F(I)
0110 IF X(I)=A+(I-1)*D THEN 140
0120 PRINT "ERROR:UNEQUALLY SPACED DATA"
0130 GOTO 9999
0140 NEXT I
0150 DISP "TOLERANCE";
0160 INPUT E
0170 DISP "# OF HARMONICS";
0180 INPUT M
0190 LET X=FNA(M,E,N)
0200 IF X=1 THEN 280
0210 PRINT TAB(10);"COEFFICIENTS OF FOURIER EXPANSION"
0220 PRINT
0230 PRINT TAB(14),"A-VECTOR            B-VECTOR        L.SQ.ERROR"
0240 FOR I=1 TO P1 STEP 1
0250 PRINT I-1,G(I),H(I),E(I)
0260 NEXT I
0270 GOTO 9999
0280 PRINT " TOO MUCH HARMONICS : M=";M
0290 GOTO 170

END OF LISTING


LINK *SLFTRP,A
9999 END
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
                DATA  POINTS VALUES

 1                0                0
 2                1                1
 3                2                2
 4                3                3
 5                4                4
 6                5                5
 7                6                4
 8                7                3
 9                8                2
 10               9                1
 11               10               0
 12               11              -1
 13               12              -2
 14               13              -3
 15               14              -4
 16               15              -5
 17               16              -4
 18               17              -3
 19               18              -2
 20               19              -1
TOLERANCE?
0
# OF HARMONICS?
15
 TOO MUCH HARMONICS : M= 15
# OF HARMONICS?
10
         COEFFICIENTS OF FOURIER EXPANSION

             A-VECTOR            B-VECTOR          L.SQ.ERROR
0                0                 0                 170
1               -5.0000000E-11     4.0863458         3.0177785
2                0                 0                 3.0177785
3                9.0000000E-13    -.48518400         .66374335
4                0                 0                 .66374335
5                1.1200000E-12     .20000000         .26374335
6               -9.8277907E-14    -1.2363735E-12     .26374335
7               -1.6800000E-12    -.12596162         .10508006
8                0                 0                 .10508006
9                2.4000000E-12     .10250856         2.8000000E-03
10               0                -1.1000000E-12     2.8000000E-09
```

Ao	Bo	N	X_1	Y_1	W_1	X_2	Y_2	W_2	- - - -

File ZZZZ

Ao first point on data file

Bo last point on data file

N number of points on data file

X_i x value of point i i = 1, 2, . . . N

Y_i y value of point i

W_i weight of point i

```
EXE FLPRINT,ZZZZ
 0                 19                20                0                 0
 1                 1                 1                 1                 2
 2                 1                 3                 3                 1
 4                 4                 1                 5                 5
 1                 6                 4                 1                 7
 3                 1                 8                 2                 1
 9                 1                 1                 10                0
 1                 11                -1                1                 12
-2                 1                 13                -3                1
 14                -4                1                 15                -5
 1                 16                -4                1                 17
-3                 1                 18                -2                1
 19                -1                1                 0                 0
READY
```

Title — Fourier interpolation

Purpose — Evaluation of the Fourier approximating expansion

Method — Known the coefficients of the Fourier expansion approximating a function (see *SLFTRP), this expansion is calculated for a set of arguments (table) with a fixed increment:

a1, a1+d, a,+2d, ... a,+kd $\leqslant$ b1

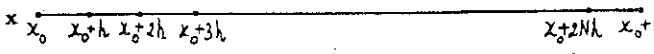
If a = b and d = 1 the expansion is evaluated for a given point only.

The arguments may be entered in various ways: in radians, degree and in the original units or with unit spacing.

The relationship between the various scales may be seen as follows:

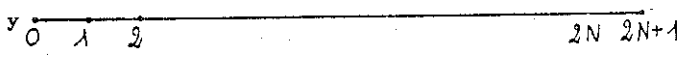
a) odd number (2N+1) of observations over $(0, 2\pi)$

. original scale

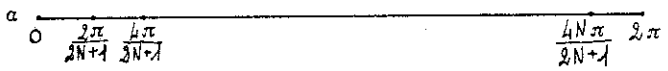


where h = space between the observations (see *SLFTRP)

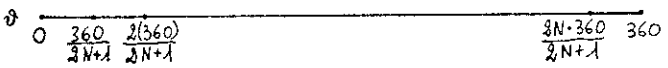
. unit spacing



. radians

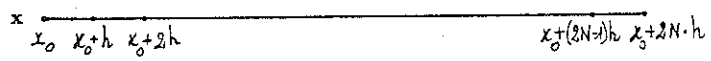


. degrees

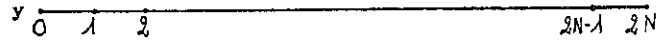


b) even number (2N) of observations over $(0, 2\pi)$

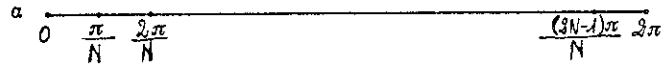
. original scale



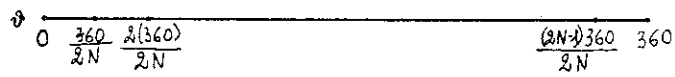
. unit spacing



. radians



degrees



Calling Statement F = FNB (T, Ao, D_1, A_1, B_1, H_1, N, M)

Parameters

T scale code:
- 1 = original
- 2 = unit spacing
- 3 = radians
- 4 = degrees

Ao x-value of first observation

D_1 space between two successive observations

A_1 lower limit of set of arguments

B_1 upper limit of set of arguments

H_1 increment

N number of observations

M number of harmonics of Fourier expansion

Global variables

- input	G ( )	Fourier coefficients for cosines	(one dimension)
	H ( )	Fourier coefficients for sines	(one dimension)
- return	O ( )	array of arguments	(one dimension)
	P ( )	Fourier expansion values	(one dimension)
	P2	number of calculated arguments	

Status value ∅

Note - The subroutine use the function FNC

Listing

```
LIST
FILE    *SLFOUI

0002 DEF FNB(T,A,D,A1,B1,H1,N,M)X
0004 LET P2=0
0006 FOR X=A1 TO B1 STEP H1
0008 LET P2=P2+1
0010 ON T GOTO 12,16,20,24
0012 LET P(P2)=FNC(X,N,M)
0014 GOTO 26
0016 LET P(P2)=FNC(X-A/D,N,M)
0018 GOTO 26
```

```
0020 LET P(P2)=FNC(X*N/(2*PI),N,M)
0022 GOTO 26
0024 LET P(P2)=FNC(X*N/360,N,M)
0026 LET Q(P2)=X
0028 NEXT X
0030 LET FN*=0
0032 FNEND
0034 DEF FNC(X,N,M)C,I,P
0036 LET P=N/2
0038 IF N-INT(N/2)*2=0 THEN 42
0040 LET P=(N-1)/2
0042 LET C=G(1)/2
0044 FOR I=1 TO M+1 STEP 1
0046 IF ABS(N-INT(N/2)*2)+ABS(I-P-1)<>0 THEN 52
0048 LET C=C+G(I)/2*COS(PI*X)
0050 GOTO 58
0052 LET C=C+G(I)*COS((I-1)*2*PI/N*X)
0054 LET C=C+H(I)*SIN((I-1)*2*PI/N*X)
0056 NEXT I
0058 LET FN*=C
0060 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RRFOUI

0010 FILES ZZZZ
0020 DIM G(21),H(21),E(21),F(21),X(21),Q(50),P(50)
0030 SETW :1 TO 1
0040 READ :1,A,B,N
0050 LET D=(B-A)/(N-1)
0060 PRINT TAB(15)," DATA  POINTS VALUES"
0070 PRINT
0080 FOR I=1 TO N STEP 1
0090 READ :1,X(I),F(I),W
0100 PRINT I,X(I),F(I)
0110 IF X(I)=A+(I-1)*D THEN 140
0120 PRINT "ERROR:UNEQUALLY SPACED DATA"
0130 GOTO 9999
0140 NEXT I
0150 DISP "TOLERANCE";
0160 INPUT E
0170 DISP "# OF HARMONICS";
0180 INPUT M
0190 LET X=FNA(M,E,N)
0200 IF X=1 THEN 280
0210 PRINT TAB(10);"COEFFICIENTS OF FOURIER EXPANSION"
0220 PRINT
0230 PRINT TAB(14),"A-VECTOR           B-VECTOR        L.SQ.ERROR"
0240 FOR I=1 TO P1 STEP 1
0250 PRINT I-1,G(I),H(I),E(I)
0260 NEXT I
0270 GOTO 300
0280 PRINT " TOO MUCH HARMONICS : M=";M
0290 GOTO 170
0300 PRINT
0310 PRINT TAB(20),"FOURIER INTERPOLATION"
0320 PRINT
0330 DISP "POINT=1,TABLE=2,STOP=0";
```

```
0340 INPUT R
0345 IF R=0 THEN 9999
0350 DISP "INTERPOLATION CODE(1-4)";
0360 INPUT T
0370 ON R+1 GOTO 9999,380,430
0380 DISP "ENTER X-VALUE";
0390 INPUT A1
0400 LET B1=A1
0410 LET H1=1
0420 GOTO 470
0430 DISP "ENTER START,END & INCR.";
0440 INPUT A1,B1,H1
0450 PRINT TAB(20);"INTERPOLATION INTERVAL=[";A1;"-";B1;"]"
0460 PRINT
0470 DISP " FOURIER INTERP. IS RUNNING"
0475 LET Y=FNB(T,A,D,A1,B1,H1,N,P1)
0480 FOR I=1 TO P2 STEP 1
0490 PRINT G(I),P(I)
0500 NEXT I
0510 GOTO 330

END OF LISTING


LINK *SLFTRP,A
LINK *SLFOUI,B
9999 END



RUN
**** FORMALLY CORRECT PROGRAM ****
                DATA  POINTS VALUES

 1                0                0
 2                1                1
 3                2                2
 4                3                3
 5                4                4
 6                5                5
 7                6                4
 8                7                3
 9                8                2
 10               9                1
 11               10               0
 12               11              -1
 13               12              -2
 14               13              -3
 15               14              -4
 16               15              -5
 17               16              -4
 18               17              -3
 19               18              -2
 20               19              -1
TOLERANCE?
0
# OF HARMONICS?
10
```

COEFFICIENTS OF FOURIER EXPANSION

	A-VECTOR	B-VECTOR	L.SQ.ERROR
0	0	0	170
1	-5.0000000E-11	4.0863458	3.0177785
2	0	0	3.0177785
3	9.0000000E-13	-.48518400	.66374335
4	0	0	.66374335
5	1.1200000E-12	.20000000	.26374335
6	-9.8277907E-14	-1.2363735E-12	.26374335
7	-1.6900000E-12	-.12596162	.10508006
8	0	0	.10508006
9	2.4000000E-12	.10250856	2.8000000E-09
10	0	-1.1000000E-12	2.8000000E-09

FOURIER INTERPOLATION

```
POINT=1,TABLE=2,STOP=0?
1
INTERPOLATION CODE(1-4)?
1
ENTER X-VALUE?
0
 FOURIER INTERP. IS RUNNING
 0              -4.7368278E-11
POINT=1,TABLE=2,STOP=0?
1
INTERPOLATION CODE(1-4)?
2
ENTER X-VALUE?
2
 FOURIER INTERP. IS RUNNING
 2               2.0000000.
POINT=1,TABLE=2,STOP=0?
1
INTERPOLATION CODE(1-4)?
3
ENTER X-VALUE?
3.14159
 FOURIER INTERP. IS RUNNING
 3.14159         9.7430583E-06
POINT=1,TABLE=2,STOP=0?
1
INTERPOLATION CODE(1-4)?
4
ENTER X-VALUE?
90
 FOURIER INTERP. IS RUNNING
 90              5.0000000
POINT=1,TABLE=2,STOP=0?
2
INTERPOLATION CODE(1-4)?
2
ENTER START,END & INCR.?
0,20,.5
```

INTERPOLATION INTERVAL=[0 - 20]

FOURIER INTERP. IS RUNNING

0	-4.7368278E-11
.5	.54941164
1	1.0000000
1.5	1.4457419
2	2.0000000
2.5	2.5665374
3	3.0000000
3.5	3.4044902
4	4.0000000
4.5	4.6829808
5	5.0000000
5.5	4.6829808
6	4.0000000
6.5	3.4044902
7	3.0000000
7.5	2.5665374
8	2.0000000
8.5	1.4457419
9	1.0000000
9.5	.54941164
10	4.7171722E-11
10.5	-.54941164
11	-1.0000000
11.5	-1.4457419
12	-2.0000000
12.5	-2.5665374
13	-3.0000000
13.5	-3.4044902
14	-4.0000000
14.5	-4.6829808
15	-5.0000000
15.5	-4.6829808
16	-4.0000000
16.5	-3.4044902
17	-3.0000000
17.5	-2.5665374
18	-2.0000000
18.5	-1.4457419
19	-1.0000000
19.5	-.54941164
20	-4.7368278E-11

*SLLLSQ

Title

Least squares curve fitting to user sepplied basis function

Purpose

To determine the combinations coefficients of an approximating function, which is a linear combination of user supplied basis functions. The standard deviation is also computed

Method

Let (x_i, y_i) $i = 1,2 \ldots N$ a given set of data, which does not exactly represent the underlying function f (x). The problem is to minimize

$$E = \sum_{i=1}^{N} (y_i - g_k(x_i))^2$$

where

$$g_k(x) = \sum_{j=1}^{k} a_j f_j(x) \qquad k \leq 6$$

is the approximating function of f (x) and fj (x) is a user supplied basis.

The determination of the a_j's is by solution of the normal equations by Crout's method.

There is no limit to the size of the data sample, but at most 6 basis functions may be used.

A common method to minimize E is to set equal to zero the partial derivates of E with respect to parameters a_j. In this way one obtains a linear system of K equation in the K unkowns $a_1, a_2, \ldots a_K$

$$\sum_{j=1}^{K} a_j \left(\sum_{i=1}^{N} f_j(x_i) f_1(x_i) \right) = \sum_{i=1}^{N} y_i f_1(x_i)$$

$$\sum_{j=1}^{K} a_j \left(\sum_{i=1}^{N} f_j(x_i) f_2(x_i) \right) = \sum_{i=1}^{N} y_i f_2(x_i)$$

$$\sum_{j=1}^{K} a_j \left(\sum_{i=1}^{N} f_j(x_i) f_K(x_i) \right) = \sum_{i=1}^{N} y_i f_K(x_i)$$

This system, which has the coefficient matrix symmetric, is solved by the Crout algorithm.

When the system is solved, it is necessary to decide if the set of found coefficients a_i $i = 1,2 \ldots K$ gives a good approximation of the function f. A measure of this is the standard deviation or tolerance defined by

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}\left[f(x_i)\right]^2 - \sum_{i=1}^{N}\left[\sum_{j=1}^{K} a_j f_j(x_i)\right]}{N}}$$

By considering successive values of σ at each step, it is possible to evaluate the effect of the addiction of each new function. The coefficients a_i are computed to index K ($K \leqslant 6$ = number of basis functions) or until σ is less than the given tolerance. If the defined basis contains two or more functions linear dependent or if one or more function are equal zero, the subroutine execution is interrupted (status value = 3) because in this case the linear system cannot be solved: it is therefore necessary to change the basis.

Calling Statement: F = FNA (N1, M1, SØ)

Parameters:

N1 number of data points
M1 number of basis functions
SØ tolerance

Global variables

- input

E ( ) x-data values (one dimension)
F ( ) y-data values (one dimension)

- work

Q (6,6) normal matrix coefficients (used in system resolution)
R (6,6) normal matrix coefficients
H (6) right-side in normal equations
G (6) idem (used in systel resolution)
P (6) control vector

-return

O (6) basis coefficients
P1 standard deviation
P2 number of used functions

Note- The basis function f_i (x) are defined as elements of an array. The following steps are necessary to fixe a basis:

- call the user program
- link this program with the subourtine (LINK *SLLLSQ,A)
- fixe the basis with these statements:
  574 G (1) = 1th basis function with variable x expressed as E (I)
  576 G (2) = 2nd basis function with variable x expressed as E (I)

578 G (3) = 3rd basis function with variable x expressed as E (I)

580 G (4) = 4th basis function with variable x expressed as E (I)

582 G (5) = 5th basis function with variable x expressed as E (I)

584 G (6) = 6th basis function with variable x expressed as E (I)

- run the program

Status values

0 = reached tolerance

1 = not reached tolerance

2 = data points number < basis functions number

3 = the normal system is singular (change basis)

Listing

```
LIST
FILE    *SLLLSQ

0002 DEF FNA(N1,M1,S0)T,N,A,K,I,J,D2,D3,S,E,L,M
0004 IF N1>=M1 THEN 10
0006 LET FN*=2
0008 GOTO 228
0010 LET T=0
0012 FOR N=1 TO M1 STEP 1
0014 IF N<>M1 THEN 18
0016 LET T=1
0018 LET A=0
0020 FOR K=1 TO M1 STEP 1
0022 LET R(K,N)=0
0024 NEXT K
0026 LET H(N)=0
0028 FOR I=1 TO N1 STEP 1
0030 LET A=A+F(I)*F(I)
0032 GOSUB 212
0034 FOR K=1 TO N STEP 1
0036 LET R(K,N)=R(K,N)+G(K)*G(N)
0038 NEXT K
0040 LET H(N)=H(N)+F(I)*G(N)
0042 NEXT I
0044 FOR I=1 TO N STEP 1
0046 LET Q(I)=H(I)
0048 LET P(I)=I
0050 FOR J=1 TO I STEP 1
0052 LET Q(I,J)=Q(J,I)=R(J,I)
0054 NEXT J
0056 NEXT I
0058 IF N<>1 THEN 70
0060 IF Q(1,1)<>0 THEN 66
0062 LET FN*=3
0064 GOTO 228
0066 LET Q(P(1))=Q(P(1))/Q(1,1)
0068 GOTO 168
0070 FOR I=1 TO N STEP 1
0072 IF I=1 THEN 90
0074 FOR J=I TO N STEP 1
0076 LET S=0
0078 FOR K=1 TO I-1 STEP 1
0080 LET S=S+Q(P(J),K)*Q(P(K),I)
0082 NEXT K
0084 LET Q(P(J),I)=Q(P(J),I)-S
0086 NEXT J
0088 IF I=N THEN 134
0090 LET E=ABS(Q(P(I),I))
0092 LET L=P(I)
0094 LET M=I
0096 FOR J=I+1 TO N STEP 1
0098 IF ABS(Q(P(J),I))<=E THEN 106
```

```
0100 LET E=ABS(Q(P(J),I))
0102 LET L=P(J)
0104 LET M=J
0106 NEXT J
0108 IF E<>0 THEN 114
0110 LET FN*=3
0112 GOTO 228
0114 LET P(M)=P(I)
0116 LET P(I)=L
0118 FOR J=I+1 TO N STEP 1
0120 LET S=0
0122 IF I=1 THEN 130
0124 FOR K=1 TO I-1 STEP 1
0126 LET S=S+Q(P(I),K)*Q(P(K),J)
0128 NEXT K
0130 LET Q(P(I),J)=(Q(P(I),J)-S)/Q(P(I),I)
0132 NEXT J
0134 IF Q(P(N),N)<>0 THEN 140
0136 LET FN*=3
0138 GOTO 228
0140 LET S=0
0142 IF I=1 THEN 150
0144 FOR K=1 TO I-1 STEP 1
0146 LET S=S+Q(P(I),K)*O(P(K))
0148 NEXT K
0150 LET O(P(I))=(O(P(I))-S)/Q(P(I),I)
0152 NEXT I
0154 FOR I=N-1 TO 1 STEP -1
0156 LET S=0
0158 FOR K=I+1 TO N STEP 1
0160 LET S=S+Q(P(I),K)*O(P(K))
0162 NEXT K
0164 LET O(P(I))=O(P(I))-S
0166 NEXT I
0168 GOSUB 192
0170 IF SQR(ABS(A-D3)/N1)>S0 THEN 180
0172 LET FN*=0
0174 LET P1=SQR(ABS(A-D3)/N1)
0176 LET P2=N
0178 GOTO 228
0180 IF T=0 THEN 190
0182 LET FN*=1
0184 LET P1=SQR(ABS(A-D3)/N1)
0186 LET P2=N
0188 GOTO 228
0190 NEXT N
0192 LET D3=0
0194 FOR I=1 TO N1 STEP 1
0196 GOSUB 212
0198 LET D2=0
0200 FOR H=1 TO N STEP 1
0202 LET D2=D2+O(P(H))*G(H)
0204 NEXT H
0206 LET D3=D3+D2*D2
0208 NEXT I
0210 RETURN
0212 REM BASIS*****
0214 LET G(1)=1
0216 LET G(2)=E(I)
0218 LET G(3)=E(I)↑2
0220 LET G(4)=E(I)↑3
0222 LET G(5)=E(I)↑4
0224 LET G(6)=E(I)↑5
0226 RETURN
0228 FNEND

END OF LISTING
```

Example of use

```
LIST
FILE     RRLLSQ

0010 DCL S (M1,T,N1,N,K,I,J,M)
0020 :     ###     #####.#####      #####.#####
0030 FILES WWWWW
0040 DIM Q(6,6),R(6,6),H(6),G(6),O(6),P(6),E(150),F(150)
0050 PRINT
0060 DISP "ENTER # OF BASIS FUNCTIONS";
0070 INPUT M1
0080 DISP "ENTER TOLERANCE";
0090 INPUT S0
0100 SETW :1 TO 5
0110 READ :1,N1
0120 PRINT "NUMBER OF POINTS  =";N1
0130 PRINT
0140 PRINT TAB(20);"LIST OF DATA POINTS"
0150 FOR I=1 TO N1 STEP 1
0160 READ :1,X,Y,W
0170 LET E(I)=X
0180 LET F(I)=Y
0190 PRINT USING 20,I,E(I),F(I)
0200 NEXT I
0205 DISP " PROGRAM IS RUNNING"
0210 LET E=FNA(N1,M1,S0)
0220 ON E+1 GOSUB 290,310,330,350
0224 PRINT
0226 PRINT TAB(20);"BASIS COEFFICIENTS"
0228 PRINT
0230 FOR I=1 TO P2 STEP 1
0240 PRINT I,O(P(I))
0250 NEXT I
0255 PRINT
0260 PRINT "ST.  DEVIATION           =";P1
0270 PRINT "NUMBER OF USED FUNCTIONS =";P2
0280 GOTO 9999
0290 PRINT "REACHED TOLERANCE"
0300 RETURN
0310 PRINT "NOT REACHED TOLERANCE"
0320 RETURN
0330 PRINT "#POINTS < # FUNCTIONS"
0340 GOTO 9999
0350 PRINT "CHANGE BASIS"
0360 GOTO 9999

END OF LISTING


LINK *SLLLSQ,A
9999 END
```

```
574 G(1)=1
576G(2)=E(I)
578G(3)=E(I)↑2
580G(4)=E(I)↑3
582G(5)=E(I)↑4
584G(6)=E(I)↑5
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****

ENTER # OF BASIS FUNCTIONS?
6
ENTER TOLERANCE?
0
NUMBER OF POINTS  = 6

                  LIST OF DATA POINTS
        1          1.10000          3.00000
        2          2.30000          0.00000
        3          3.00000          4.00000
        4          4.50000         -3.00000
        5          6.00000        -12.00000
        6          9.00000        456.00000
 PROGRAM IS RUNNING
NOT REACHED TOLERANCE

                  BASIS COEFFICIENTS

 1               64.611439
 2              -103.93260
 3               55.435628
 4              -11.744731
 5               .87329265
 6              -5.6100513E-03

ST.  DEVIATION           = 2.0534524E-03
NUMBER OF USED FUNCTIONS = 6
```

Ao	Bo	N	X_1	Y_1	W_1	X_2	Y_2	W_2	- - - -

File WWWWW

Ao first point on data file

Bo last point on data file

N number of data points

X_i x-value of point i i = 1, 2, . . . N

Y_i y-value of point i

W_i weight of point i

```
EXE FLPRINT,WWWWW
 1.1              3                6                1.1              3
 1                2.3              0                1                3
 4                1                4.5             -3                1
 6               -12               1                9              456
 1                0                0                0                0
 0                0                0                0                0
 0                0                0                0                0
 0                0                0                0                0
 0                0                0                0                0
```

*SLNLLS

Title

Non linear least squares curve fitting to an arbitrary scalar function

Purpose

To calculate a least squares approximation to a given model with six or less parameters

Method

The choosen method is the Powell Algorithm improved by Zangwill.
Let (x_i, y_i) $i = 1,2 \ldots N$ a given set of data points.
The subroutine computes a least square fit of these data to the model $f(a_1, a_2 \ldots a_K, X)$ $K \leqslant 6$.
That is, let

$$f = \sum_{i=1}^{N} \left[y_i - f(a_1, a_2 \ldots a_K, x_i) \right]^2$$

the function that defines the difference squared of the set of data to the model $f(a_1, a_2, .., a_K, X)$, one determines $a_1, a_2 \ldots a_K$ so that the function f is minimized.
The $f(a_1, a_2 \ldots a_K, X)$ is defined by user in his calling program as FNA (X).
Let C_r, $r = 1,2 \ldots n$ be the coordinate directions and assume they are normalized to unit length.
Initialization step: let an initial point p_n^0, and n normalized directions ξ_i^1 , $r = 1,2 \ldots n$ be given.
Calculate λ_n^0 to minimize $f(p_n^0 + \lambda_n^0 \xi_n^1)$ and let $p_{n+1}^0 = p_n^0 + \lambda_n^0 \xi_n^1$
Set $t = 1$ and go to iteration K with $K = 1$.
Iteration K : p_{n+1}^{K-1}, ξ_r^K, $r = 1,2 \ldots n$ and t are given.
Step (i): Find α to minimize $f(p_{n+1}^{K-1} + \alpha C_t)$. Update by

$$t \leftarrow \begin{cases} t + 1 & \text{if } 1 \leqslant t < n \\ 1 & \text{if } t = n. \end{cases}$$

If $\alpha \neq 0$, let $p_0^K = p_{n+1}^{K-1} + \alpha C_t$. If $\alpha = 0$, repeat step (i).
Should step (i) be repeated n times in succession. Stop: the point p_{n+1}^{k-1} is optimal.
Step (ii): For $r = 1, \ldots, n$ calculate λ_r^K, to minimize

$$f(p_{r-1}^K + \lambda_r^K \xi_r^K)$$

and define

$$p_r^K = p_{r-1}^K + \lambda_r^K \xi_r^K$$

Let

$$\xi_{n+1}^K = (p_n^K - p_{n+1}^{K-1}) / \| p_n^K - p_{n+1}^{K-1} \|$$

Determine λ^k_{n+1} to minimize $f(p^k_n + \lambda^k_{n+1}\,\xi^k_{n+1})$

and set $p^k_{n+1} = p^k_n + \lambda^k_{n+1}\,\xi^k_{n+1}$

define $\xi^{k+1}_t = \xi^k_{t+1}$ $\quad t = 1, \ldots n$

Go to iteration k with k + 1 replacing k.

Some discussion of the procedure may be in order.

Step (i) proceeds cyclically through the coordinate directions. That is, each time we return to step (i) we use the next coordinate direction, repeating C_1 after using C_n. Every n + 1 times step (i) is employed the same coordinate direction is employed. The t indexes the coordinate direction to be used. If step (i) is repeated n times in succession, then all n coordinate direction have been attempted and no change in the point has occurred. Such a situation can only occur if at that point the gradient of the function f is zero. As we assume f strictly convex and continuously differentiable, that point is optimal.

In general step (i) is repeated until a new point is generated. In step (ii) the procedure continues as in the earlier procedures. It is important to observe that after at most n iteration, all coordinate directions have been used.

Both in step (i) where we check if the found point is a minimum and in step (ii), it is necessary a routine by which to minimize the function along a line.

To find the minimum on a line, the following data must be provided:

(i) a point on the line, p

(ii) the direction of the line, ξ

(iii) an upper bound to the length of step along the line, m

(iv) the length of step along the line, q, assumes to be less than m, and

(v) the accuracy to which the minimum is required, e.

The method of minimization must be such as to find the minimum of a quadratic form, so it is primarily based on the quadratic defined by three function values.

Initially f(p) and f(p + q ξ) are calculated, and then either f(p - q ξ) or f(p + 2q ξ) is worked out depending on whether f(p) is less than or greater than f(p + q ξ).

There three function values are now used in the general formula which predicts the turning value of the quadratic defined by a, f(p + a ξ), b, f(p + b ξ), c and f(p + c ξ) to be at (p + d ξ), where

$$d = \frac{1}{2} \cdot \frac{(b^2-c^2)f_a+(c^2-a^2)f_b+(a_2-b_2)f_c}{(b-c)f_a+(c-a)f_b+(a-b)f_c}$$

It is a minimum if

$$\frac{(b-c)f_a+(c-a)f_b+(a-b)f_c}{(a-b)(b-c)(c-a)} < 0$$

If the turning value is predicted to be a maximum, or if the value of d is such that to calculate f(p + d ξ) a step greater than m must be taken, the maximum allowed step is taken in the direction of decreasing f, and the function value at the point is furthest from (p + d ξ) is discarted, so the prediction may be repeated. Otherwise d is compared with a, b and c, and, if it is within the required accuracy of one of them, that point is chosen as the minimum. If it is not, f(p + d ξ) is calculated so that the quadratic prediction may be repeated; the function value which is thrown away out of f(p + a ξ), f(p + b ξ) and f(p + c ξ) is normally the greatest, but it is not if rejecting a smaller one can yield a definite bracket on a minimum, which would not be obtained otherwise.

At the end of the program the minimum reached indicates the value of the set of parameters satisfying the least squares fit of given (x_i, y_i) i = 1,2 ... N in the model.

Calling Statement

F = FNB (NØ)

Parameters

N Ø = number of directiones

Global variables

- input

O Tolerance
O2 max number of iterationes
O4 step for minimization along line
P (NØ) parameters of minimization vector

- return

P1 = P (1)
P2 = P (2)
P3 = P (3)
P4 = P (4)
P5 = P (5)
P6 = P (6)

} final set of parameters

	Q	generical residual value given by residual calculation subroutine
	R9	status value
- work	G (6)	vector used to store intermediate result during minimization
	R (6,7)	direction matrix
	Q (6)	initial values of parameters during minimization along a line
	O (6)	intermediate values of parameters during minimization along a line
	H (6)	present minimization directions
	QØ	previous minimum residual value
	Q1	residual value in 1st abscissa in minimum along line
	Q2	residual value in 2nd abscissa in minimum along line
	Q3	residual value in 3rd abscissa in minimum along line
	Q4	minimum residual value
	Q9	residual value of Q (I)

Status value	Ø	correct calculation
	1	too much iterations (no solution yet after the fixed iterations)

Note - The subroutine uses the functions FNF and FND. The user model f (a_1 a_2 ...a_k, X) with K < 6 is defined as FNA (X) in the calling program and the parameters are indicated with the letters P_1 P_2... P_k. For example, to define $a_1 + a_2$ a_3 x modified exponential curve, the following statement is used:

```
FNA =  P1 + P2 * P3 ↑ X
```

Listing

```
LIST
FILE    *SLNLLS

0002 DEF FNB(N0)T,S,I,J,M,M2,F,B0,A,B,C,D,T1,M1
0004 LET R9=0
0006 FOR J=1 TO N0 STEP 1
0008 FOR I=1 TO N0 STEP 1
0010 IF I=J THEN 16
0012 LET R(I,J)=0
0014 GOTO 18
0016 LET R(I,J)=1
0018 NEXT I
0020 NEXT J
0022 FOR I=1 TO N0 STEP 1
0024 LET H(I)=R(I,N0)
0026 NEXT I
0028 GOSUB 120
0030 IF R9=1 THEN 530
0032 LET T=2
0034 FOR I=1 TO N0 STEP 1
0036 LET G(I)=P(I)
0038 NEXT I
```

```
0040 LET S=1
0042 FOR I=1 TO N0 STEP 1
0044 IF I=T THEN 50
0046 LET H(I)=0
0048 GOTO 52
0050 LET H(I)=1
0052 LET P(I)=G(I)
0054 NEXT I
0056 GOSUB 120
0058 IF R9=1 THEN 530
0060 LET T=T+1
0062 IF T<N0+1 THEN 66
0064 LET T=1
0066 IF ABS(D)>0 THEN 74
0068 LET S=S+1
0070 IF S>=N0+1 THEN 530
0072 GOTO 42
0074 LET S=1
0076 FOR J=1 TO N0 STEP 1
0078 FOR I=1 TO N0 STEP 1
0080 LET H(I)=R(I,J)
0082 NEXT I
0084 GOSUB 120
0086 IF R9=1 THEN 530
0088 NEXT J
0090 FOR I=1 TO N0 STEP 1
0092 LET H(I)=P(I)-G(I)
0094 NEXT I
0096 GOSUB 512
0098 GOSUB 120
0100 IF R9=1 THEN 530
0102 FOR J=1 TO N0-1 STEP 1
0104 FOR I=1 TO N0 STEP 1
0106 LET R(I,J)=R(I,J+1)
0108 NEXT I
0110 NEXT J
0112 FOR I=1 TO N0 STEP 1
0114 LET R(I,N0)=H(I)
0116 NEXT I
0118 GOTO 34
0120 REM *****MINIMUM ALONG A LINE CALC. SUBR.
0122 DISP "RESIDUAL ITERATION";Q2
0124 LET Q2=Q2-1
0126 IF Q2>=0 THEN 130
0128 LET R9=1
0130 LET F=Q4
0132 LET M2=M=10*Q4
0134 LET B0=1E99
0136 LET T1=1
0138 GOSUB 484
0140 LET Q1=Q3=Q0=0
0142 LET M1=A=0
0144 FOR I=1 TO N0 STEP 1
0146 LET Q(I)=P(I)
0148 NEXT I
0150 LET M1=A
0152 LET C=B=F
0154 FOR I=1 TO N0 STEP 1
0156 LET Q(I)=Q(I)
0158 LET P(I)=Q(I)+B*H(I)
0160 NEXT I
0162 GOSUB 484
0164 LET Q2=Q3=Q
0166 IF M2>0 THEN 194
0168 IF ABS(Q1-Q2)<1E-4 THEN 208
0170 IF Q2<Q1 THEN 208
0172 FOR I=1 TO N0 STEP 1
0174 LET Q(I)=P(I)
0176 NEXT I
0178 LET B=Q4
0180 LET Q2=Q1
0182 LET M1=A=0
0184 LET Q1=Q3
0186 LET M2=C=2*Q4
0188 LET Q3=Q4
```

```
0190 LET T1=2
0192 GOTO 334
0194 IF ABS(Q1-Q2)<1E-4 THEN 208
0196 IF Q1>Q2 THEN 208
0198 LET F=-F
0200 LET M2=-M2
0202 LET D=B
0204 LET Q4=Q2
0206 GOTO 152
0208 LET C=2*F
0210 IF ABS(C)<=ABS(M2) THEN 214
0212 LET C=M2
0214 FOR I=1 TO N0 STEP 1
0216 LET P(I)=O(I)+C*H(I)
0218 NEXT I
0220 GOSUB 484
0222 LET Q3=Q
0224 LET D=FND(A,B,C,Q1,Q2,Q3)
0226 IF FNF(A,B,C,Q1,Q2,Q3)>=0 THEN 230
0228 IF ABS(D)<ABS(M2) THEN 232
0230 LET D=M2
0232 FOR I=1 TO N0 STEP 1
0234 LET P(I)=O(I)+D*H(I)
0236 NEXT I
0238 GOSUB 484
0240 LET Q4=Q
0242 IF ABS(Q4-Q0)>Q/10 THEN 258
0244 IF ABS(Q4-Q1)>1E-4 THEN 432
0246 IF A<>0 THEN 432
0248 IF M2<0 THEN 432
0250 LET F=-F
0252 LET M2=-M2
0254 LET Q1=Q9
0256 GOTO 150
0258 LET Q0=Q4
0260 IF ABS(D-B)<1E-3 THEN 378
0262 IF ABS(D-C)<1E-3 THEN 378
0264 IF ABS(D)>ABS(C) THEN 278
0266 IF Q4<Q3 THEN 292
0268 LET M1=A=B
0270 LET Q1=Q2
0272 LET B=D
0274 LET Q2=Q4
0276 GOTO 224
0278 LET M1=A=B
0280 LET Q1=Q2
0282 LET B=C
0284 LET Q2=Q3
0286 LET C=D
0288 LET Q3=Q4
0290 IF Q3<Q2 THEN 224
0292 LET M2=C
0294 LET T1=2
0296 IF ABS(D-B)<1E-3 THEN 378
0298 ON SGN(ABS(D)-ABS(B))+2 GOTO 300,378,318
0300 IF Q4>=Q2 THEN 312
0302 LET C=B
0304 LET Q3=Q2
0306 LET B=D
0308 LET Q2=Q4
0310 GOTO 334
0312 LET A=D
0314 LET Q1=Q4
0316 GOTO 334
0318 IF Q4>=Q2 THEN 330
0320 LET A=B
0322 LET Q1=Q2
0324 LET B=D
0326 LET Q2=Q4
0328 GOTO 334
0330 LET C=D
0332 LET Q3=Q4
0334 LET D=FND(A,B,C,Q1,Q2,Q3)
0336 FOR I=1 TO N0 STEP 1
```

```
0338 LET P(I)=Q(I)+D*H(I)
0340 NEXT I
0342 GOSUB 484
0344 LET Q4=Q
0346 IF ABS(A-D)<1E-3 THEN 378
0348 IF ABS(C-D)<1E-3 THEN 378
0350 IF (D-M1)*(M2-D)<=0 THEN 360
0352 IF FNF(A,B,C,Q1,Q2,Q3)>=0 THEN 360
0354 IF ABS(Q4-Q0)<Q/10 THEN 450
0356 LET Q0=Q4
0358 GOTO 296
0360 LET B=(A+C)/2
0362 IF B=B0 THEN 432
0364 LET B0=B
0366 FOR I=1 TO N0 STEP 1
0368 LET P(I)=Q(I)+B*H(I)
0370 NEXT I
0372 GOSUB 484
0374 LET Q2=Q
0376 ON T1 GOTO 224,334
0378 FOR I=1 TO N0 STEP 1
0380 LET P(I)=Q(I)+(D-.01)*H(I)
0382 NEXT I
0384 GOSUB 484
0386 IF Q>Q4 THEN 402
0388 IF M2<0 THEN 396
0390 LET C=D-.01
0392 LET Q3=Q
0394 GOTO 360
0396 LET A=D-.01
0398 LET Q1=Q
0400 ON T1 GOTO 150,360
0402 FOR I=1 TO N0 STEP 1
0404 LET P(I)=Q(I)+(D+.01)*H(I)
0406 NEXT I
0408 GOSUB 484
0410 FOR I=1 TO N0 STEP 1
0412 LET P(I)=Q(I)+D*H(I)
0414 NEXT I
0416 IF Q>Q4 THEN 450
0418 IF M2<0 THEN 426
0420 LET A=D+.01
0422 LET Q1=Q
0424 ON T1 GOTO 150,360
0426 LET C=D+.01
0428 LET Q3=Q
0430 GOTO 360
0432 IF Q4<=Q1 THEN 438
0434 LET Q4=Q1
0436 LET D=A
0438 IF Q4<=Q2 THEN 444
0440 LET Q4=Q2
0442 LET D=B
0444 IF Q4<=Q3 THEN 450
0446 LET Q4=Q3
0448 LET D=C
0450 REM ***** OUTPUT
0452 FOR I=1 TO N0 STEP 1
0454 LET P(I)=Q(I)+D*H(I)
0456 NEXT I
0458 LET D=0
0460 GOSUB 484
0462 IF Q<Q9 THEN 474
0464 FOR I=1 TO N0 STEP 1
0466 LET P(I)=Q(I)
0468 NEXT I
0470 LET Q=Q9
0472 GOTO 482
0474 FOR I=1 TO N0 STEP 1
0476 LET D=D+(P(I)-Q(I))*(P(I)-Q(I))
0478 NEXT I
0480 LET D=SQR(D)
0482 RETURN
```

```
0484 REM *****FUNCT. CALC. IN EACH POINT SUBR.
0486 LET Q=0
0488 LET P1=P(1)
0490 LET P2=P(2)
0492 LET P3=P(3)
0494 LET P4=P(4)
0496 LET P5=P(5)
0498 LET P6=P(6)
0500 FOR I=1 TO N STEP 1
0502 LET X=E(I)
0504 LET Y=F(I)
0506 LET Q=Q+(Y-FNA(X))*(Y-FNA(X))
0508 NEXT I
0510 RETURN
0512 REM ********** NORMALIZATION SUBROUTINE *****
0514 LET J=0
0516 FOR I=1 TO N0 STEP 1
0518 LET J=J+H(I)*H(I)
0520 NEXT I
0522 FOR I=1 TO N0 STEP 1
0524 LET H(I)=H(I)/SQR(J)
0526 NEXT I
0528 RETURN
0530 LET FN*=R9
0532 FNEND
0534 DEF FND(A,B,C,Q1,Q2,Q3)H
0536 LET H=(B*B-C*C)*Q1+(C*C-A*A)*Q2+(A*A-B*B)*Q3
0538 IF ABS((B-C)*Q1+(C-A)*Q2+(A-B)*Q3)>1E-10 THEN 544
0540 LET FN*=1E99
0542 GOTO 548
0544 LET H=H/(2*((B-C)*Q1+(C-A)*Q2+(A-B)*Q3))
0546 LET FN*=H
0548 FNEND
0550 DEF FNF(A,B,C,Q1,Q2,Q3)=((B-C)*Q1+(C-A)*Q2+(A-B)*Q3)/((A-B)*(B-C)*(C-A))

END OF LISTING
```

Example of use

```
LIST
FILE     RRNLLS

0010 DEF FNA(X)=P1+P2*P3↑X
0020 FILES AAAAAA
0030 DIM G(6),R(6,7),P(6),Q(6),O(6),H(6),E(30),F(30)
0040 SETN :1 TO 5
0050 READ :1,N
0060 FOR I=1 TO N STEP 1
0070 READ :1,X,Y,W
0080 LET E(I)=X
0090 LET F(I)=Y
0100 PRINT E(I),F(I)
0110 NEXT I
0120 DISP "ENTER # OF PARAMETERS (N0)";
0130 INPUT N0
0140 PRINT
0150 DISP "ENTER INITIAL SET OF PARAMETERS"
0160 DELAY 20
0170 FOR I=1 TO N0 STEP 1
0180 DISP "ENTER P(";I;") =";
0190 INPUT P(I)
0200 NEXT I
0210 FOR I=N0+1 TO 6 STEP 1
0220 LET P(I)=0
0230 NEXT I
0240 DISP "ENTER TOLERANCE";
0250 INPUT O
0260 DISP "ENTER STEP FOR LINE MINIMIZATION";
0270 INPUT O4
0280 DISP "ENTER MAX NUMBER OF ITERATIONS";
0290 INPUT O2
0300 LET R9=FNB(N0)
0310 PRINT TAB(20);"FINAL SET OF PARAMETERS"
0320 FOR I=1 TO N0 STEP 1
0330 PRINT I,P(I)
0340 NEXT I
0350 PRINT
0360 PRINT "RESIDUAL VALUE =";Q
0370 IF R9=0 THEN 9999
0380 PRINT "TOO MUCH ITERATIONS"
0390 GOTO 9999

END OF LISTING


LINK *SLNLLS,B
9999 END
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
 .2                24.3
 .5                23.4
 .7                22.8
 .8                22.5
 1.2               21.5
 1.4               21.1
 1.7               20.5
 1.9               20.1
 2.3               19.4
 2.5               19.1
 2.7               18.8
 3                 18.4
 3.3               18.1
 3.6               17.8
 3.9               17.5
 4.1               17.3
 4.4               17.1
 4.7               16.9
 5                 16.7
 5.3               16.5
 5.7               16.3
 6.2               16.1
 6.7               15.9
 7.3               15.7
ENTER # OF PARAMETERS (N0)?
3

ENTER INITIAL SET OF PARAMETERS
ENTER P( 1 ) =?
10
ENTER P( 2 ) =?
10
ENTER P( 3 ) =?
1
ENTER TOLERANCE?
.001
ENTER STEP'FOR LINE MINIMIZATION?
.5
ENTER MAX NUMBER OF ITERATIONS?
100
RESIDUAL ITERATION 100


                  FINAL SET OF PARAMETERS
 1               14.965961
 2               10.029389
 3               .70222642

RESIDUAL VALUE = 1.0430485E-02
```

Title

Fitting to a sum of exponentials. Prony's method

Purpose

To evaluate the exponents b_i and the factors a_i of a sum of exponentials of the form:

$$f(x) = a_1 e^{b1\,x} + a_2 e^{b2\,x} + \ldots + a_m e^{bm\,x}$$

wath exactly fits the data points

Method

Let (x_i, y_i) $i = 2,4,6$ or 8 be a given set of equally spaced data points, the routine calculates a sum of exponentials of the form:

$$f(x) = a_1 e^{b1\,x} + a_2 e^{b2\,x} + \ldots + a_m e^{bm\,x}$$

$$n = 2m \qquad n = 2,4,6,8$$

such that $y_i = f(x_i)$.

Prony's method is based on the fact that for evenly spaced data each $e^{bk\,x}$ $K = 1,2, \ldots m$ satisfies some fixed m-th order difference equation with constant coefficients whose characteristic equation has root $r_k = e^{bk}$.

To summarize the calculation, a system of m linear equations in m-unknowns must be solved. This system is derived from the difference equation which requires that the data be equally spaced and an even number of points x_j $(j = 1,2, \ldots 2m)$.

The number of points must equal the number of exponentials plus the number of coefficients.

Thus with the results of solved system, a polynomial of degree m is obtained and all the roots of this polynomial (the characteristic equation) must be found.

Recall that $r_k = e^{bk}$, where r_k are the roots of the characteristic equation, it is straight forward to explicitly calculate the b_k.

Three cases must be considered:

1) r_k = positive real number, then $b_k = \log(r_k)$

2) r_k = negative real number.

Then let $r_k = -\varrho_k$ $\varrho_k > 0$ and

$$e^{b_k x} = r_k^x = \varrho_k^x (-1)^x = e^{\log(\varrho_k) x} \cos(\pi x)$$

3) r_k is complex, then $r_k = r_{k+1}$ is also a root.

Let $r_k = a + ib$ then $r_{k+1} = a - ib$

$$\varrho_k = \sqrt{a^2+b^2} \qquad \vartheta_k = \tan^{-1}(b/a)$$

Then:

$$e^{b_k x} = r_k^x = (a+ib)^x = \varrho_k^x(\cos\vartheta x + i \sin\vartheta x)$$

$$e^{b_{k+1} x} = r_{k+1}^x = (a - b)^x = \varrho_k^x(\cos\vartheta x - i \sin\vartheta x)$$

In this last case, the presence of complex value is momentary. The corresponding coefficients a_k and a_{k+1} must be complex coniugates, i.e. for real some a'_k, a'_{k+1}

$$a_K = \frac{a'_k + a'_{k+1} i}{2} \qquad a_{k+1} = \frac{a'_k - a'_{k+1}}{2}$$

then

$$a_k e^{b_k x} + a_{k+1} e^{b_{k+1} x} = a'_k e^{\log(\varrho_k)x} \cos\vartheta + a'_{k+1} e^{\log(\varrho_k)x} \sin\vartheta$$

The exponents having been calculated, the problem becomes one of solving m linear equation for the m unknowns $a_1, a_2 \ldots a_m$.

Since there not exists a formula for the exact solution of an arbitrary polynomial of degree greater than 4, a suitable restriction for the number of data points has been made ($n \leq 8$).

After the coefficients and exponents have been calculated, the user can evaluate the function point by point or for a table of values, using the following sequence:

```
J = (X-A)/ H + 1

F = 0

FOR I = 1 TO N1/2

F = F + Q (I) * EXP ( (J-1)* G(I)) *COS (H(I) * (J-1) + R(I))
NEXT I
```

where X = point to be interpolated
N1 = number of data points
H = space between the data points
A = first data point
F = y - found value
for the array see the output global variables.

Calling Statement F = FNA (N1)

Parameter N1 = number of points (2,4,6,8)

Global variables

-Input
- O ( ) y-values of data points (one dimension)

-Return
- Q ( ) coefficients (one dimension)
- G ( ) exponents
- H ( ) factors (to calculate the cos argument)
- R ( ) factors (to calculate the cos argument)
- E ( ) found values for the data points

Note The elements of R ( ) are $\emptyset$ or $\pm \pi/2$

-Work
- A (4,4) matrix of linear system
- P (5) array of index
- F (5)
- PØ + P5, Q1 + Q5, R1 + R4 } service variables

Function value
- Ø correct calculation
- 1 linear system is singular (No solution is possible)

References:

1. Numerical Methods for Scientist and Engineers Richard Hanning - M. Graw Hill 1972 pp. 617-627

2. Handbook of Numerical Method and Applications Louis Kelley - Addison - Wesley 1967 pp. 80-82.

Listing

```
OLD *SLPRON
LIS
FILE     *SLPRON


0002 DEF FNA(N1)T1,I,J,K,S,E,L,N,U,V,T,T4,T2,T3
0004 DIM G(5),H(5),Q(5),P(5),B(4,4),F(5),R(5)
0006 LET T1=N=N1+2
0008 FOR I=1 TO N STEP 1
0010 FOR J=1 TO N STEP 1
0012 LET A(I,J)=0(I+J-1)
0014 NEXT J
0016 LET Q(I)=-I(N-I)
0018 LET P(I)=I
0020 NEXT I
0022 GOSUB 102
0024 FOR I=1 TO N STEP
0026 LET E(I)=Q(P(I))
0028 NEXT I
0030 LET E(N+1)=1
0032 GOTO 200
0034 LET I=1
0036 IF ABS(H(I))>0 THEN 56
0038 IF G(I)<0 THEN 48
0040 LET G(I)=LOG(G(I))
0042 LET R(I)=H(I)=0
0044 LET I=I+1
0046 ON SGN(I-N)+2 GOTO 36,36,72
0048 LET G(I)=LOG(-G(I))
0050 LET H(I)=PI
0052 LET R(I)=0
0054 GOTO 44
0056 LET U=SQR(G(I)*G(I)+H(I)*H(I))
0058 LET V=ATN(H(I)/G(I))+(1-SGN(G(I)))*PI/2
0060 LET G(I)=G(I+1)=LOG(U)
0062 LET H(I)=H(I+1)=V
0064 LET R(I)=0
0066 LET R(I+1)=-PI/2
0068 LET I=I+2
0070 ON SGN(I-N)+2 GOTO 32,36,72
0072 FOR J=1 TO N STEP 1
0074 LET S(J)=0(J)
0076 FOR I=1 TO N STEP 1
0078 LET A(J,I)=EXP((J-1)*G(I))*COS(H(I)*(J-1)+R(I))
0080 NEXT I
0081 NEXT J
0084 GOSUB 102
0086 FOR J=1 TO N+2 STEP 1
0088 LET E(J)=0
0090 FOR I=1 TO N STEP 1
0092 LET E(J)=E(J)+Q(P(I))*EXP((J-1)*G(I))*COS(H(I)*(J-1)+R(I))
0094 NEXT I
0096 NEXT J
0098 LET FN+=0
0100 GOTO 330
0102 FOR I=1 TO N STEP 1
0104 IF I=1 THEN 122
0106 FOR J=I TO N STEP 1
0108 LET S=0
0110 FOR K=1 TO I-1 STEP 1
0112 LET S=S+A(P(J),K)*A(P(K),I)
0114 NEXT K
0116 LET A(P(J),I)=A(P(J),I)-S
0118 NEXT J
0120 IF I=N THEN 166
```

```
0122 LET E=ABS(A(P(I),I))
0124 LET L=P(I)
0126 LET M=I
0128 FOR J=I+1 TO N STEP 1
0130 IF ABS(A(P(J),I))<=E THEN 138
0132 LET E=ABS(A(P(J),I))
0134 LET L=P(J)
0136 LET M=J
0138 NEXT J
0140 IF E<>0 THEN 146
0142 LET FN=1
0144 GOTO 390
0146 LET P(M)=P(I)
0148 LET P(I)=L
0150 FOR J=I+1 TO N STEP 1
0152 LET S=0
0154 IF I=1 THEN 162
0156 FOR K=1 TO I-1 STEP 1
0158 LET S=S+A(P(I),K)*A(P(K),J)
0160 NEXT K
0162 LET A(P(I),J)=(A(P(I),J)-S)/A(P(I),I)
0164 NEXT J
0166 IF A(P(N),N)<>0 THEN 170
0168 GOTO 142
0170 LET S=0
0172 IF I=1 THEN 180
0174 FOR K=1 TO I-1 STEP 1
0176 LET S=S+A(P(I),K)*Q(P(K))
0178 NEXT K
0180 LET Q(P(I))=(Q(P(I))-S)/A(P(I),I)
0182 NEXT I
0184 FOR I=N-1 TO 1 STEP -1
0186 LET S=0
0188 FOR K=I+1 TO N STEP 1
0190 LET S=S+A(P(I),K)*Q(P(K))
0192 NEXT K
0194 LET Q(P(I))=Q(P(I))-S
0196 NEXT I
0198 RETURN
0200 LET Y=0
0202 ON T1 GOTO 204,212,216,290
0204 LET Q(X+1)=-E(2)/E(1)
0206 LET H(K+1)=0
0208 LET K=K+1
0210 GOTO 34
0212 LET T=0
0214 GOTO 242
0216 LET T=E(2)^2+E(4)
0218 LET T1=(E(2)-E(3)+T)/(3*E(4))
0220 LET T2=(T*(E(2)-2*E(4)*T*T)-E(1))/(2*E(4))
0222 LET T4=T2^3+T^2
0224 IF T2<>0 THEN 226
0226 IF T2<>0 THEN 232
0228 LET R2=0
0230 GOTO 248
0232 LET R3=SGN(T3)*EXP(LOG(ABS(T3)+1E-99)/3)
0234 GOTO 248
0236 IF T4<=0 THEN 260
0238 LET R1=T3+SQR(T4)
0240 LET R1=SGN(R1)*EXP(LOG(ABS(R1)+1E-99)/3)
0242 LET R2=T3-SQR(T4)
0244 LET R2=SGN(R2)*EXP(LOG(ABS(R2)+1E-99)/3)
0246 LET R3=R1+R2
0248 LET R4=R3-T
0250 IF T1=4 THEN 306
0252 LET Q(K+1)=54
0254 LET H(K+1)=0
```

```
0256 LET K=K+1
0258 GOTO 280
0260 LET R1=T3/SQR(ABS(T2*T2*T2))
0262 LET R2=PI/2
0264 IF R1=0 THEN 268
0266 LET R2=ATN(SQR(ABS(1-R1*R1))/R1)+.5*(1-SGN(R1))*PI
0268 LET R3=2*SQR(ABS(T2))*COS(R2/3)
0270 LET R4=R3-T
0272 IF T1=4 THEN 306
0274 LET [illegible]
0276 LET H(K+1)=0
0278 LET K=K+1
0280 LET E(4)=0
0282 LET E(3)=1
0284 LET E(2)=R3
0286 LET E(1)=R3*R3+3*T2
0288 GOTO 342
0290 FOR I=1 TO 5 STEP 1
0292 LET F(I)=E(I)/E(5)
0294 NEXT I
0296 LET E(4)=1.0
0298 LET E(3)=-F(3)
0300 LET E(2)=F(4)*F(2)-4*F(1)
0302 LET E(1)=4*F(3)*F(1)-F(4)*F(4)*F(1)-F(2)*F(2)
0304 GOTO 215
0306 LET T=0
0308 LET P3=F(4)*F(4)/4+R4-F(3)
0310 LET P4=P5=F(4)/2
0312 IF ABS(P3)<1E-9 THEN 324
0314 LET P4=P4+SQR(P3)
0316 LET P5=P5-SQR(P3)
0318 LET Q4=(F(2)+P4*R4*P5-P4*F(3))/(P5-P4)
0320 LET Q5=F(3)-P4*P5-Q4
0322 GOTO 328
0324 LET Q4=R4/2+SQR(R4*R4/4-F(1))
0326 LET Q5=R4/2-SQR(R4*R4/4-F(1))
0328 LET E(3)=1.0
0330 LET E(2)=P4
0332 LET E(1)=Q4
0334 GOTO 342
0336 LET T1=2
0338 LET E(2)=P5
0340 LET E(1)=Q5
0342 LET P0=E(2)*E(2)-4*E(3)*E(1)
0344 ON SGN(P0)+2 GOTO 360,346,352
0346 LET P1=P2=-E(2)/(2*E(3))
0348 LET P3=0
0350 GOTO 364
0352 LET P1=(-E(2)+SQR(P0))/(2*E(3))
0354 LET P2=(-E(2)-SQR(P0))/(2*E(3))
0356 LET P3=0
0358 GOTO 364
0360 LET P1=P2=-E(2)/(2*E(3))
0362 LET P3=SQR(ABS(P0))/(2*E(3))
0364 IF P3<>0 THEN 378
0366 LET G(K+1)=P1-T
0368 LET G(K+2)=P2-T
0370 LET H(K+2)=H(K+1)=0
0372 LET K=K+2
0374 IF T1=4 THEN 336
0376 GOTO 34
0378 LET G(K+2)=G(K+1)=P1-T
0380 LET H(K+1)=P3
0382 LET H(K+2)=-P3
0384 LET K=K+2
0386 IF T1=4 THEN 336
0388 GOTO 34
0390 FNEND

END OF LISTING
```

Example of use

```
OLD FRPRON
LIS
FILE     FRPRON

0010 DIM O(8),Q(5),H(5),E(8),Q(5),P(5),A(4,4),F(5),R(5)
0020 FILES (0,0)
0030 PRINT
0040 SETW 1 TO 1
0050 READ 1;A,B,N1
0060 IF (N1-2)*(N1-4)*(N1-5)*(N1-8)=0 THEN 80
0070 PRINT "THERE MUST BE 2,4,5 OR 8 PAIRS ONLY"
0080 GOTO 9999
0090 LET H=(B-A)/(N1-1)
0100 PRINT TAB(20);"LIST OF DATA"
0110 FOR I=1 TO N1 STEP 1
0120 READ 1;O(I),W
0130 PRINT O(I)
0140 IF I=1 THEN 180
0150 IF W<>0 THEN 180
0160 PRINT "UNEQUALLY SPACED DATA"
0170 GOTO 9999
0180 LET W=1
0190 NEXT I
0200 LET E=FNA(N1)
0210 PRINT
0220 PRINT "COEFFICIENTS";TAB(20);"EXPONENTS";TAB(45);"FACTORS"
0230 PRINT
0240 FOR I=1 TO N STEP 1
0250 PRINT A(I,1);TAB(20);A(I,2);TAB(38);A(I,3);TAB(56);P(I)
0260 NEXT I
0270 PRINT
0280 PRINT "EVALUATION"
0290 PRINT "X-VALUE","Y-VALUE","EXPECTED VALUE"
0300 PRINT "-------","-------","--------------"
0310 FOR J=1 TO N1 STEP 1
0320 PRINT A+H*(J-1),O(J),E(J)
0330 NEXT J
0340 GOTO 9999

END OF LISTING
```

```
LINK *SLPRON,A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

                    LIST OF DATA
 0                  0
 1                  1
 2                  4
 3                  9
 4                  16
 5                  25
 6                  36
 7                  49

COEFFICIENTS        EXPONENTS              FACTORS

 83741061.          -1.3228732E-05     0              0
-1.1525173E-05       .40546511         3.1415927      0
-83741051.           4.4514373E-05     7.3260710E-05  0
 1.1509342E-05       4.4514373E-05     7.7253710E-05 -1.5707963

EVALUATION
X-VALUE             Y-VALUE          FOUND VALUE
-------             -------          -----------
 0                  0.               -.00001
 1                  1                 1.0000044
 2                  4                 3.9999595
 3                  9                 9.0000354
 4                  16                15.999342
 5                  25                24.999979
 6                  36                35.999748
 7                  49                49.000055
```

Title Lagrange - Aitken interpolation

Purpose To interpolate a given set of data to an order (the maximum degree of the polynomial used) and accuracy specified by the user

Method Whereas the strict Lagrange interpolation method uses all N points of a given data set to find a polynomial P(x) of degree (N-1) which fits the data exactly, the Lagrange Aitken method generates interpolating polynomials of successively higher degree from the polynomial of lower degree.
In particular, let (x_i, y_i) i = 1,2 ... N a given set if data points, if $\tilde{x}$ is the value to be interpolated, the points (x_i, y_i) and (x_{i+1}, y_{i+1}) near $\tilde{x}$ are used to determine a linear interpolating polynomial P1 (x). Next the point (x_{i+2}, y_{i+2}) near $\tilde{x}$ is used along with the previous to determine a quadratic interpolating polynomial P2 (x).
Then the point (x_{i+3}, y_{i+3}) near $\tilde{x}$ is used to determine a cubic interpolating polynomial P3 (x) and so on.
The polynomial of degree d+1 is generated using d+2 points.
The polynomial coefficients can be computed explicitly by the following algorithm:

Aitken Algorithm

(1) $P_{K,0}(x) = y_K \qquad K = 0,1,2 \ldots N-1$

(2) $$P_{K,d+1}(x) = \frac{(x_K - x)\, P_{d,d}(x) - (x_d - x)\, P_{K,d}(x)}{x_K - x_d}$$
$$K = d+1, \ldots N-1$$

Equation (2) calculates a polynomial of degree d+1 from two polynomial of degree d. The use of this algorithm produces a triangular scheme :

$P_{0,0}$
$P_{1,0} \quad P_{1,1}$
$P_{2,0} \quad P_{2,1} \quad P_{2,2}$
$\vdots$
$P_{N-1,0} \; P_{N-1,1} \; \ldots\ldots \; P_{N-1,N-1}$

The diagonal elements are the Lagrange polynomials, i.e.

$P_{1,1} = P_1(x)$, $P_{2,2} = P_2(x)$ ..... $P_{N-1,N-1} = P_{N-1}(x)$.

The scheme is used to directly compute $P_1(\tilde{x})$, $P_2(\tilde{x})$ .....
Further interpolation is unnecessary when there is a very little difference between successive value.
Given a value $\tilde{x}$, a maximum order t and a tolerance E, the routine compute row-wise the Aitken's scheme. After the third row, it tests if $|P_{i+1}(\tilde{x}) - P_i(\tilde{x})| < E$. If this condition holds, further interpolation is hold to be unnecessary and $P_{i+1}(\tilde{x})$ is returned as the interpolated value.
If the condition is not satisfied after t+1 rows, then $P_t(\tilde{x})$ is taken as interpolated value.
The routine performs a search on a given data to locate a neighborhood around $\tilde{x}$.
If $\tilde{x}$ is a data point, the y-value is returned.
Note. The order of interpolation is limited to N-1 or 100 whichever is less.
The amount of calculation increases more than linearly with a increase in the order of interpolation. Even thought calculation is performed in double precision, round-off error must be kept in mind when high order interpolation is used.

Calling Statement

F = FNA (A,B,N,P,E,A1,B1,H1)

Parameters

A = x_1 first point of the given set
B = x_N last point of the given set
N = number of data points
P = maximum degree $\left\{ \begin{array}{l} \leqslant N-1 \text{ or } 100 \\ \text{whichever is less} \end{array} \right.$
E = tolerance (accuracy)
A1 = lower limit of table to interpolate
B1 = upper limit of table to interpolate
H1 = increment for table

Note. If A1 = B1 and H1 = 1, the interpolation is point by point.

Global variables

- Input

E ( ) x-value of data points (one dimension)
F ( ) y-value of data points

- Return

P2 number of interpolated points
P ( ) x-value of interpolated points

	Q ( )	y-value of interpolated points
-Work	X ( )	x-value of used data point (one dimension)
	Y ( )	y-value of successive Lagrangian polynomials
Status value	Ø	Complete calculation
	1	Two x-values are the same . Only (P2-1) points have been interpolated (interpolation partially executed).

Listing

```
OLD *SLAITK
LIST
FILE    *SLAITK

0002 DEF FNA(A,B,N,P,E,A1,B1,H)J,J1,I9,I1,I2,I3,I4
0004 DIM X(101),Y(101)
0005 LET FN=0
0007 LET P1=0
0010 FOR I9=A1 TO B1 STEP H1
0012 LET P2=P2+1
0014 IF I9.=A THEN 20
0016 LET J=1
0018 GOTO 72
0020 ON SGN(I9-B)+2 GOTO 32,22,29
0022 LET P(P2)=E(N)
0024 LET Q(P2)=F(N)
0026 GOTO 68
0027 LET J=N-P
0030 GOTO 72
0032 LET I1=E(1)
0034 LET I2=F(1)
0036 LET J1=1
0038 LET I3=E(J1+1)
0040 LET I4=F(J1+1)
0042 ON SGN((I1-I9)*(I3-I9))+2 GOTO 54,44,56
0044 IF I1=I9 THEN 58
```

```
0046 LET I1=I3
0048 LET I2=I4
0050 LET P(P2)=I1
0052 LET Q(P2)=I2
0054 GOTO 88
0056 LET I1=I3
0058 LET I2=I4
0060 LET J1=J1+1
0062 GOTO 78
0064 IF J1>=INT((P+1)/2) THEN 70
0066 LET J=1
0068 GOTO 72
0070 LET J=N-P
0072 FOR I1=1 TO P+1 STEP 1
0074 LET X(I1)=S(J)
0076 LET Y(I1)=F(J)
0078 LET J=J+1
0080 NEXT I1
0082 GOSUB 90
0084 LET P(P2)=I9
0086 NEXT I9
0088 GOTO 124
0090 FOR J=2 TO P+1 STEP 1
0092 LET I1=J-1
0094 FOR I=1 TO I1 STEP 1
0096 LET I2=X(I)-X(J)
0098 IF I2<>0 THEN 106
0100 LET FN+=1
0102 LET I9=81
0104 GOTO 122
0106 LET Y(J)=Q(I)+(I9-X(J))*(F(J)+(I3-X(I))) I2
0108 NEXT I
0110 LET I3=ABS(Y(J)-Y(J-1))
0112 IF J<3 THEN 116
0114 IF I3<E THEN 120
0116 NEXT J
0118 LET J=J-1
0120 LET Q(P2)=Y(J)
0122 RETURN
0124 FNEND

END OF LISTING
```

Example of use

```
OLD AAAITK
LIS
FILE    AAAITK

0020 FILES 0000
0030 PRINT
0040 DIM E(100),F(100),P(50),I(50)
0120 SETW #1 TO 1
0130 READ #1,A,B,N
0140 LET P1=N-1
0145 PRINT TAB(20),"LIST OF DATA POINTS"
0150 FOR I=1 TO N STEP 1
0160 READ #1,E(I),F(I),W
0170 PRINT E(I),F(I)
0180 NEXT I
0190 IF P1<100 THEN 210
0200 LET P1=100
0210 DISP "ENTER DEGREE (1 TO";P1;")"
0220 INPUT P
0230 IF (P-1)*(P-P1)<=0 THEN 270
0240 DISP "ERROR--"
0250 GOTO 210
0270 IF P-INT(P)=0 THEN 320
0280 DISP "ERROR--INTEGER VALUES ONLY    ++          "
0290 GOTO 210
0310 DISP "ENTER TOLERANCE "
0320 INPUT E
0340 IF E>0 THEN 370
0350 DISP "ERROR--POSITIVE VALUES ONLY   ++
0360 GOTO 320
0370 DISP "INTERP.CODE POINT=1,TABLE=2"
0380 INPUT F
0390 IF (F-1)*(F-2)=0 THEN 430
0400 DISP "ERROR INTERP. CODE"
0410 DELAY 20
0420 GOTO 370
0430 PRINT
0470 PRINT "TOLERANCE =";E
0480 PRINT "DEGREE    =" P
0490 ON F GOTO 660,500
0500 DISP "ENTER START END & INCR.".
0510 INPUT A1,B1,H1
0520 IF (A-A1)*(A1-B)<0 THEN 550
0530 IF (A-B1)*(B1-B)>=0 THEN 590
0540 DISP "TABLE";A1;"-";B1;"OUTSIDE THE INTERVAL"
0570 DELAY 20
0580 GOTO 500
0590 IF A1<=B1 THEN 630
0600 LET L9=A1
0610 LET A1=B1
0620 LET B1=L9
0630 IF H1>0 THEN 750
0640 DISP "ERROR-INCR.MUST BE POSITIVE"
0650 DELAY 10
0660 GOTO 500
0670 GOTO 760
0680 DISP "ENTER X-VALUE";
0690 INPUT A1
0700 IF (A-A1)*(A1-B)>=0 THEN 740
0710 PRINT "X =";A1;" ";"OUTSIDE THE INTERVAL"
0720 DELAY 10
0730 GOTO 680
0740 LET B1=A1
0750 LET H1=1
0760 LET Y=FNA(A,B,N,P,E,A1,B1,H1)
0770 ON Y+1 GOTO 780,890
```

```
0760 ON F GOTO 780,930
0780 PRINT "X [illegible]
0800 PRINT "ANOTHER INTERP NO=0,YES=1"
0810 INPUT G
0820 ON G+1 GOTO 9999,210
0830 PRINT TAB(20);"INTERPOLATED POINTS"
0840 PRINT TAB(8);"X-VALUE         Y-VALUE"
0850 FOR I=1 TO P2 STEP 1
0860 PRINT P(I),Q(I)
0870 NEXT I
0880 GOTO 800
0890 PRINT "TWO X-VALUES ARE THE SAME"
0900 ON F GOTO 800,910
0910 LET P2=P2-1
0920 IF P2=0 THEN 9999
0930 PRINT
0940 PRINT "INTERPOLATION PARTIALLY EXECUTED"
0950 PRINT
0960 GOTO 830

END OF LISTING


LINK *ELAITK.B
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

                    LIST OF DATA POINTS
 0                    0
 1                    1
 2                    4
 3                    9
 4                    16
 5                    25
 6                    36
 7                    49
ENTER DEGREE (1 TO 7 )?
5
ENTER TOLERANCE ?
1E-6
INTERP.CODE POINT=1 TABLE=2?
1

TOLERANCE = .000001
DEGREE      = 5
ENTER X-VALUE?
3.2
X   = 3.2              Y   = 10.24
ANOTHER INTERP NO=0,YES=1?
1
ENTER DEGREE (1 TO 7 )?
6
ENTER TOLERANCE ?
1E-6
INTERP.CODE.POINT=1,TABLE=2?
2

TOLERANCE = .000001
DEGREE      = 6
ENTER START.END & INCR.?
2,5,.25
                    INTERPOLATED POINTS
        X-VALUE         Y-VALUE
 2                    4
 2.25                 5.0625
 2.5                  6.25
 2.75                 7.5625
 3                    9
 3.25                 10.5625
 3.5                  12.25
 3.75                 14.0625
 4                    16
 4.25                 18.0625
 4.5                  20.25
 4.75                 22.5625
 5                    25
ANOTHER INTERP NO=0,YES=1?
0
```

Title Least-squares smoothing degree 0,1,2

Purpose To perform a least-squares smoothing of a set of data points; degrees 0,1 or 2 smoothing are available.

Method Let (x_i,y_i) i = 1,2 ... N be a set of given data and f (x) the underlying function from which the data have been derived.

A least squares approximation is, in general, used when the data cannot be taken as exact. The aim of smoothing is to try to reduce the "noise" present in the data by using a polynomial of low degree which will approximate the data well, but not so accuratly so as to include the "noise".

The approximation is made using only a few points in the neighborhood of x and not all the data points.

Let:

K = degree of smoothing K = 0,1,2

$f1(x) = 1$ $f2(x) = x$ $f3(x) = x^2$

x^* a point in the data sample

$g(x,a) = \sum_{i=1}^{K+1} a_i f_i(x)$ the function approximating f(x) in the least squares sense.

The coefficients a_i, ... a_{K+1} are to be choosen so as to minimize the error function:

$$E^*(x^*, a_1, \ldots, a_{K+1}) = \sum [y - g(x,a)]^2$$

where M are a set of points in the neighborhood of x^*.

For each point x^*, this minimization problem is performed; the smoothed value is $g(x^*,a)$.

For degree Ø smoothing, the polynomials is a_1. A least squares approximation is made for the x_j (j = 2,3,...,N-1) and for each x_j the points x_{j-1}, x_j, x_{j+1} are in M.

For x_1 and x_N, the approximating functions at x_2 and x_{N-1} respectively are used.

For degree 1, $f_1(x) = 1$ and $f_2(x) = x$ are used. For each point x_j (j = 2,3,..., N-1) a polynomial $a_2 x + a_1$ is the approximating function in the least squares sense and the points x_{j-1}, x_j, x_{j+1} are in M. For x_1 and x_N, the approximating functions at x_2 and

x_N respectively are used.
For degree 2, $f_1(x) = 1$ $f_2(x) = x$ $f_3(x) = x^2$ are used.
For each point x_j (j = 3,4,...,N-2) a polynomial of the form $a_3 x^2 + a_2 x + a_1$ is the approximating function in the least squares sense and x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+2} are in M.
For x_1 and x_2 the function at x_3 is used. For x_{N-1} and x_N the function at x_{N-2} is used.

Calling Statement F = FNA (M,N)

Parameters
M smoothing degree
N number of data points

Global variables

-Input
E ( ) x-values of data points (one dimension)
F ( ) y-values of data points

-Return
G ( ) smoothed-values

-Work
H (3,3) matrix of system by which the coefficients a_1, ... a_{K+1} are determined (K = smoothing degree)

Note. The user must dimension G (as E and F) in his program.

Status value
Ø correct calculation
1 not enough points in the sample
2 elements not in order ($x_{j+1} \leq x_j$)
3 matrix Hill -conditioned.

Listing

```
OLD *ELSM00
LIE
FILE     *SLSM00


0002 DEF FNA(N,M)C1,C2,C3,D1,D2,D3,I,A,B,D,I1,F
0004 DIM H(3,3)
0006 LET FN=0
0008 IF N>=5 THEN 18
0010 IF (N-2)*(N-5)<0 THEN 16
0012 LET FN=1
0014 GOTO 220
0016 IF M=1 THEN 12
0018 ON M+1 GOTO 20,42,92
0020 IF E(2)<E(1) THEN 25
0022 LET FN=2
0024 GOTO 220
0026 IF E(3)<=E(2) THEN 22
0028 LET G(1)=G(2)=(F(1)+F(2)+F(3))/3
0030 FOR I=1 TO N-3 STEP 1
0032 IF E(I+3)<=E(I+2) THEN 22
0034 LET G(I+2)=(F(I+1)+F(I+2)+F(I+3))/3
```

```
0036 NEXT I
0038 LET G(N)=(F(N-2)+F(N-1)+F(N))/3
0040 GOTO 220
0042 IF E(2)<=E(1) THEN 22
0044 IF E(3)<=E(2) THEN 22
0046 LET I1=0
0048 GOSUB 58
0050 LET G(1)=A+B+E(1)
0052 LET G(2)=A+B+E(2)
0054 FOR I1=1 TO N-3 STEP 1
0056 IF E(I1+2)<=E(I1+2) THEN 22
0058 GOSUB 58
0060 LET G(I1+2)=A+B+E(I1+2)
0062 NEXT I1
0064 LET G(N)=A+B+E(N)
0066 GOTO 220
0068 LET C1=C2=H(1,2)=H(2,2)=0
0070 FOR I=1 TO 3 STEP 1
0072 LET C1=C1+F(I+I1)
0074 LET C2=C2+E(I+I1)*F(I+I1)
0076 LET H(1,2)=H(1,2)+E(I+I1)
0078 LET H(2,2)=H(2,2)+E(I+I1)*E(I+I1)
0080 NEXT I
0082 LET H(1,1)=3
0084 LET D=H(1,1)*H(2,2)-H(1,2)*H(1,2)
0086 LET A=(C1*H(2,2)-C2*H(1,2))/D
0088 LET B=(H(1,1)*C2-H(1,2)*C1)/D
0090 RETURN
0092 LET H(1,2)=E(1)
0094 LET H(1,3)=E(1)*E(1)
0096 LET H(2,3)=H(1,3)*E(1)
0098 LET H(3,3)=H(2,3)*E(1)
0100 LET C1=F(1)
0102 LET C2=E(1)*F(1)
0104 LET C3=E(1)*F(1)*E(1)
0106 FOR I=2 TO 5 STEP 1
0108 IF E(I)<=E(I-1) THEN 22
0110 LET H(1,2)=H(1,2)+E(I)
0112 LET H(1,3)=H(1,3)+E(I)*E(I)
0114 LET H(2,3)=H(2,3)+E(I)*E(I)*E(I)
0116 LET H(3,3)=H(3,3)+E(I)*E(I)*E(I)*E(I)
0118 LET C1=C1+F(I)
0120 LET C2=C2+E(I)*F(I)
0122 LET C3=C3+E(I)*E(I)*F(I)
0124 NEXT I
0126 LET H(1,1)=5
0128 GOSUB 184
0130 IF F=1 THEN 218
0132 FOR I=1 TO 3 STEP 1
0134 LET G(I)=D1+D2*E(I)+D3*E(I)*E(I)
0136 NEXT I
0138 FOR I1=1 TO N-5 STEP 1
0140 LET H(1,2)=H(1,2)-E(I1)
0142 LET H(1,3)=H(1,3)-E(I1)*E(I1)
0144 LET H(2,3)=H(2,3)-E(I1)*E(I1)*E(I1)
0146 LET H(3,3)=H(3,3)-E(I1)*E(I1)*E(I1)*E(I1)
0148 LET C1=C1-F(I1)
0150 LET C2=C2-E(I1)*F(I1)
0152 LET C3=C3-E(I1)*E(I1)*F(I1)
0154 LET H(1,2)=H(1,2)+E(I1+5)
0156 LET H(1,3)=H(1,3)+E(I1+5)*E(I1+5)
0158 LET H(2,3)=H(2,3)+E(I1+5)*E(I1+5)*E(I1+5)
0160 LET H(3,3)=H(3,3)+E(I1+5)*E(I1+5)*E(I1+5)*E(I1+5)
0162 LET C1=C1+F(I1+5)
0164 LET C2=C2+E(I1+5)*F(I1+5)
0166 LET C3=C3+E(I1+5)*E(I1+5)*F(I1+5)
0168 LET H(1,1)=5
0170 GOSUB 184
0172 IF F=1 THEN 218
0174 LET G(I1+3)=D1+D2*E(I1+3)+D3*E(I1+3)*E(I1+3)
0176 NEXT I1
0178 LET G(N-1)=D1+D2*E(N-1)+D3*E(N-1)*E(N-1)
0180 LET G(N)=D1+D2*E(N)+D3*E(N)*E(N)
0182 GOTO 220
0184 LET D=H(1,1)*(H(1,3)*H(3,3)-H(2,3)*H(2,3))
0186 LET D=D-H(1,2)*(H(1,2)*H(3,3)-H(1,3)*H(2,3))
0188 LET D=D+H(1,3)*(H(1,2)*H(2,3)-H(1,3)*H(1,3))
0190 IF ABS(D)>1E-20 THEN 196
```

```
0192 LET F=1
0194 GOTO 216
0196 LET D1=C1*(H(1,3)*H(3,3)-H(2,3)*H(2,2))
0198 LET D1=D1-C2*(H(1,2)*H(3,3)-H(1,3)*H(2,3))
0200 LET D1=(D1+C3*(H(1,2)*H(2,3)-H(1,3)*H(1,3)))/D
0202 LET D2=H(1,1)*(C2*H(2,3)-H(2,3)*C3)
0204 LET D2=D2-H(1,2)*(C1*H(3,3)-H(1,3)*C3)
0206 LET D2=(D2+H(1,3)*(C1*H(2,3)-H(1,3)*C2))/D
0208 LET D3=H(1,1)*(H(1,3)*C3-H(2,3)*C2)
0210 LET D3=D3-H(1,2)*(H(1,2)*C3-H(1,3)*C2)
0212 LET D3=(D3+C1*(H(1,2)*H(2,3)-H(1,3)*H(1,3)))/D
0214 LET F=0
0216 RETURN
0218 LET FN*=3
0220 FNEND

END OF LISTING
```

Example of use

```
OLD RRSM00
LIS
FILE     RRSM00


0010 FILES 2222
0020 PRINT
0030 DIM E(50),F(50),G(50)
0040 PRINT "LEAST-SQUARES SMOOTHING :DEGREE 0,1,2"
0050 PRINT
0060 DISP "SMOOTHING DEGREE(0,1,2)";
0070 INPUT M
0080 IF M<>INT(M) THEN 110
0090 IF M<0 THEN 110
0100 IF M<=2 THEN 140
0110 DISP "ERROR DEGREE"
0120 DELAY 10
0130 GOTO 60
0140 SETU :1 TO 1
0150 READ :1,A,B,N
0160 FOR I=1 TO N STEP 1
0170 READ :1,E(I),F(I),W
0180 NEXT I
0190 LET Y=FNA(M,N)
0200 ON Y+1 GOTO 210,200,290,290
0210 PRINT TAB(20);"DEGREE OF SMOOTHING IS";M
0220 PRINT
0230 PRINT "X-VALUE     ","Y-VALUE     ","SMOOTHED VALUE"
0240 PRINT "--------    ","--------    ","-------- -----"
0250 FOR I=1 TO N STEP 1
0260 PRINT E(I),F(I),G(I)
0270 NEXT I
0280 GOTO 430
0290 PRINT TAB(20);"LIST OF DATA POINTS"
0300 PRINT
0310 PRINT "X-VALUE     ","Y-VALUE     "
0320 PRINT "--------    ","--------"
0330 FOR I=1 TO N STEP 1
0340 PRINT E(I),F(I)
0350 NEXT I
0360 PRINT
0370 ON Y GOTO 380,400,420
0380 PRINT "ERROR--NOT ENOUGH POINTS"
0390 GOTO 430
0400 PRINT "ERROR--ELEMENTS NOT IN ORDER"
0410 GOTO 430
0420 PRINT "MATRIX IS ILL-CONDITIONED"
0430 DISP "ANOTHER DEGREE:YES=1,NO=0";
0440 INPUT P
0450 ON P+1 GOTO 9999,60

END OF LISTING
```

```
LINK *SLSMOO.A
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

LEAST-SQUARES SMOOTHING (DEGREE 0,1,2)

                    DEGREE OF SMOOTHING IS 1

X-VALUE          Y-VALUE          SMOOTHED VALUE
--------         --------         --------------
 0                0                0
 1                1                1
 2                2                2
 3                3                3
 4                4                4
 5                5                4.3333333
 6                4                4
 7                3                3
 8                2                2
 9                1                1
 10               0                0
 11               -1               -1
 12               -2               -2
 13               -3               -3
 14               -4               -4
 15               -5               -4.3333333
 16               -4               -4
 17               -3               -3
 18               -2               -2
 19               -1               -1

                    DEGREE OF SMOOTHING IS 2

X-VALUE          Y-VALUE          SMOOTHED VALUE
--------         --------         --------------
 0                0                0
 1                1                1
 2                2                2
 3                3                3
 4                4                4.1714286
 5                5                4.9571429
 6                4                4.1714286
 7                3                3
 8                2                2
 9                1                1
 10               0                0
 11               -1               -1
 12               -2               -2
 13               -3               -3
 14               -4               -4.1714286
 15               -5               -4.9571429
 16               -4               -4.1714286
 17               -3               -3
 18               -2               -2
 19               -1               -1

                    DEGREE OF SMOOTHING IS 0
```

X-VALUE	Y-VALUE	SMOOTHED VALUE
0	0	1
1	1	1
2	2	2
3	3	3
4	4	4
5	5	4.3333333
6	4	4
7	3	3
8	2	2
9	1	1
10	0	0
11	-1	-1
12	-2	-2
13	-3	-3
14	-4	-4
15	-5	-4.3333333
16	-4	-4
17	-3	-3
18	-2	-2
19	-1	-1

Title

Weighted least squares orthogonal polynomials curve fit. Forsythe method.

Purpose

To compute the coefficients of the Forsythe polynomials approximating a set of observed data.

Method

The Forsythe polynomials are a set of orthogonal polynomials over a discrete set of weighted data points (x_i, y_i, w_i). The algorithm for generating these polynomials is as follows:

$$
\begin{aligned}
P_{-i}(x) &= 0 \\
P_0(x) &= 1 \\
P_1(x) &= (x - U_1)\, P_0(x) \\
P_2(x) &= (x - U_2)\, P_1(x) - V_1\, P0(x) \\
&\vdots \\
P_m(x) &= (x - U_m)\, P_{m-1}(x) - V_m\, P_{m-2}(x)
\end{aligned}
$$

where

$$U_m = \frac{\sum_{j=1}^{N} x_i \, (P_{m-1}(x_i))^2 \, w_i}{D_{m-1}}$$

$$N_{m-1} = \frac{\sum_{j=1}^{N} x_i \, P_{m-1}(x_i) \, P_{m-2}(x_i) \, w_i}{D_{m-2}}$$

$$D_m = \sum_{j=1}^{N} (P_m(x_i))^2 \, w_i$$

The aim is to approximate the function f(x) by a linear combination of the polynomials

$$\text{il} \quad y(x) = \sum_{m=0}^{M} a_m P_m(x)$$

y(x) is the desired approximation. The coefficients are given by

$$a_m = \frac{\sum_{j=1}^{N} y_i P_m(x_i) w_i}{D_m}$$

It is possible to rewrite this equation in terms of the variable x as follows:

$$y(x) = \sum_{m=0}^{M} a_m P_m(x) = \sum_{K=0}^{M} c_K^M x^K$$

where $$c_K^M = \sum_{m=K}^{M} a_m b_K^m$$

and

$$b_K^m = \begin{cases} 0 & K < 0, \text{ or } K > m \\ 1 & K = m \\ b_{K-1}^{m-1} - \omega_m b_K^{m-1} - v_{m-1} b_K^{m-2} & 0 \le K < m \end{cases}$$

The subroutine computes the coefficients C_K.

Calling Statement

F = FNZ (S,M,N)

Parameters

S = tolerance
M = maximum degree of resultant polynomial
N = number of points

Global variables

Input

E ( ) array of data point abscissae (one dimension)
F ( ) array of data point ordinates (one dimension)
G ( ) array of data point weights (one dimension)

Return

P1 degree of resultant polynomial

R ( ) array polynomials coefficients, in increasing order:

$R(1) = a_0$, $R(i) = a_{i-1}$

H ( ) work array (two dimensions: 7 rows, M + 2 columns)

Status value

FNZ = Ø

Listing

```
FILE    *SLFSYT

0002 DEF FNB(S,M,P)T1,A,D2,D3,W1,S0,K,K1,K2,I,J
0004 LET T1=1
0006 LET FN*=0
0008 LET A=J=D3=D2=D1=0
0010 FOR K=1 TO 7 STEP 1
0012 LET H(K,1)=0
0014 FOR I=1 TO M+1 STEP 1
0016 LET R(I)=H(K,I+1)=0
0018 NEXT I
0020 NEXT K
0022 FOR I=1 TO P STEP 1
0024 LET A=A+F(I)*F(I)*G(I)
0026 LET H(6,3)=H(6,3)+E(I)*G(I)
0028 LET D2=D2+G(I)
0030 LET H(1,1)=H(1,1)+F(I)*G(I)
0032 NEXT I
0034 IF M<>0 THEN 38
0036 LET T1=2
0038 LET W1=D1=D2
0040 LET H(1,1)=H(1,1)/D1
0042 LET H(6,3)=H(6,3)/D1
0044 LET D3=H(1,1)*H(1,1)*D2
0046 LET S0=SQR((A-D3)/W1)
0048 IF SQR((A-D3)/W1)>S THEN 56
0050 LET T1=2
0052 GOSUB 132
0054 GOTO 182
0056 LET D2=0
0058 GOSUB 132
0060 IF T1=1 THEN 66
0062 LET FN*=1
0064 GOTO 182
0066 FOR J=1 TO M STEP 1
0068 IF J<>M THEN 72
0070 LET T1=2
0072 LET H(5,1)=0
0074 LET H(5,2)=1
0076 FOR I=1 TO P STEP 1
0078 FOR K=1 TO J STEP 1
0080 LET H(5,K+2)=(E(I)-H(6,K+2))*H(5,K+1)-H(7,K+2)*H(5,K)
0082 NEXT K
0084 IF J=M THEN 90
0086 LET H(6,J+3)=H(6,J+3)+E(I)*H(5,J+2)*H(5,J+2)*G(I)
0088 LET H(7,J+3)=H(7,J+3)+E(I)*H(5,J+2)*H(5,J+1)*G(I)
0090 LET D2=D2+H(5,J+2)*H(5,J+2)*G(I)
0092 LET H(1,J+1)=H(1,J+1)+F(I)*H(5,J+2)*G(I)
0094 NEXT I
0096 IF J=M THEN 102
0098 LET H(6,J+3)=H(6,J+3)/D2
0100 LET H(7,J+3)=H(7,J+3)/D1
```

```
0102 LET D1=D2
0104 LET H(1,J+1)=H(1,J+1)/D1
0106 LET D3=D3+H(1,J+1)*H(1,J+1)*D2
0108 IF ABS(S0-SQR(ABS(A-D3)/W1))<1E-6 THEN 114
0110 LET S0=SQR(ABS(A-D3)/W1)
0112 IF S0>S THEN 120
0114 LET T1=2
0116 GOSUB 132
0118 GOTO 182
0120 LET D2=0
0122 GOSUB 132
0124 IF T1=1 THEN 130
0126 LET FN*=1
0128 GOTO 182
0130 NEXT J
0132 IF J<>0 THEN 138
0134 LET R(1)=H(1,1)
0136 GOTO 180
0138 LET H(4,2)=H(2,2)=1
0140 LET H(3,2)=0
0142 FOR K1=2 TO J+1 STEP 1
0144 FOR K2=1 TO K1-1 STEP 1
0146 LET H(2,K2+1)=H(4,K2)-H(6,K1+1)*H(4,K2+1)-H(7,K1+1)*H(3,K2+1)
0148 NEXT K2
0150 LET H(2,K1+1)=1
0152 FOR K2=1 TO K1 STEP 1
0154 IF K2=K1 THEN 160
0156 LET H(3,K2+1)=H(4,K2+1)
0158 GOTO 162
0160 LET H(3,K2+1)=0
0162 LET H(4,K2+1)=H(2,K2+1)
0164 NEXT K2
0166 NEXT K1
0168 LET K1=J+1
0170 FOR I=1 TO K1 STEP 1
0172 LET R(I)=R(I)+H(2,I+1)*H(1,K1)
0174 NEXT I
0180 RETURN
0182 LET P1=J
0184 FNEND

END OF LISTING
```

Example of use

```
FILE     RRFSYT

0010 FILES AAAAAA
0015 PRINT
0020 DIM E(150),F(150),G(150),H(7,22),R(21)
0030 SETW :1 TO 5
0040 READ :1,N
0050 PRINT "NUMBER OF POINTS : ";N
0060 PRINT
0070 PRINT "  X","  Y","WEIGHT"
0080 FOR I=1 TO N STEP 1
0090 READ :1,E(I),F(I),G(I)
0100 PRINT E(I),F(I),G(I)
0110 NEXT I
0120 DISP "ENTER TOLERANCE";
0130 INPUT S
0140 DISP "ENTER DEGREE (MAX 20)";
0150 INPUT M
```

```
0160 DISP "ENTER X&Y SCALE FACTORS";
0170 INPUT 01, 02
0180 FOR I=1 TO N STEP 1
0190 LET E(I)=E(I)/01
0200 LET F(I)=F(I)/02
0210 NEXT I
0215 PRINT
0220 PRINT "TOLERANCE = ";S,"SCALE FACTORS = ";01;" ";02
0230 PRINT "MAXIMUM DEGREE = ";M
0240 IF FNB(S,M,N)=0 THEN 270
0250 PRINT
0260 PRINT "TOLERANCE NOT REACHED"
0270 PRINT
0280 PRINT "DEGREE OF POLY : ";P1
0290 PRINT "COEFFICIENTS (IN INCREASING ORDER)"
0300 LET H5=1
0310 FOR I=1 TO P1+1 STEP 1
0320 LET R(I)=R(I)*02/H5
0330 PRINT R(I),
0340 LET H5=H5*01
0350 NEXT I
0360 GOTO 9999

END OF LISTING


LINK *SLFSYT,B
9999 END
RUN
**** FORMALLY CORRECT PROGRAM ****

NUMBER OF POINTS :  24

  X                Y                WEIGHT
 .2               24.3               1
 .5               23.4               1
 .7               22.8               1
 .8               22.5               1
 1.2              21.5               1
 1.4              21.1               1
 1.7              20.5               1
 1.9              20.1               1
 2.3              19.4               1
 2.5              19.1               1
 2.7              18.8               1
 3                18.4               1
 3.3              18.1               1
 3.6              17.8               1
 3.9              17.5               1
 4.1              17.3               1
 4.4              17.1               1
 4.7              16.9               1
 5                16.7               1
 5.3              16.5               1
 5.7              16.3               1
 6.2              16.1               1
 6.7              15.9               1
 7.3              15.7               1

TOLERANCE =      0                  SCALE FACTORS =  1 ; 1
MAXIMUM DEGREE =  10

DEGREE OF POLY :  5
COEFFICIENTS (IN INCREASING ORDER)
 24.976952      -3.4756816       .55822106      -4.7389369E-02   1.7243481E-03
-5.4007000E-06
```

Title

Rational function fitting-Pade approximation.

Purpose

To calculate the coefficients of the rational function (a quotient of two polynomials of specified degrees) which best approximates a given function on an interval.

Method

The degree of both the numerator and denominator are specified by the calling program and must be less than 10.

Given a function $f(x)$, the routine finds polynomials $P_m(x)$ and $Q_n(x)$ so that $\frac{P_m(x)}{Q_n(x)}$ approximates $f(x)$ near $x = x_o$, where x_o is the center of the interval (m and n denote the degree of respective polynomials).

The conditions used to determine the coefficients are that the two functions $\frac{P_m(x)}{Q_n(x)}$ and $f(x)$ agree at x_o and so do the first $(m + n)$ derivatives.

The approximate

$$f(x) = \sum_{j=0}^{\infty} c_j x^j \text{ by } \frac{P_m(x)}{Q_n(x)} \text{ near } x = 0$$

where c_j are the Taylor series coefficients, and

$$P_m(x) = a_o + a_1 x + \ldots + a_m x^m$$

$$Q_n(x) = b_o + b_1 x + \ldots + b_n x^n$$

one considers

$$E(x) = \sum_{j=0}^{\infty} c_j x^j - \frac{P_m(x)}{Q_n(x)}$$

and the a's and the b's must be choosen so that $E(x)$ and its first several derivatives are equal to zero at $x = 0$.

There are $(m+n+1)$ unknown a's and b's; thus the first $(m+n)$ derivatives of $E(x)$ can be made equal to zero.

$E(x)$ can be written as follows:

$$E(x) \frac{\sum_{j=0}^{m} \sum_{k=0}^{j} (c_{j-k} b_k - a_j) x^j + \sum_{j=m+1}^{\infty} \sum_{k=0}^{j} c_{j-k} b_k x^j}{\sum_{j=0}^{n} b_j x^j}$$

```
0122 NEXT I
0124 IF N9=9 THEN 132
0126 FOR I=N9+2 TO 10 STEP 1
0128 LET G(I)=0
0130 NEXT I
0132 LET G(1)=1
0134 FOR I=N9+1 TO 1 STEP -1
0136 LET F(I)=0
0138 FOR J=1 TO I STEP 1
0140 LET F(I)=F(I)+G(J)*E(I-J+1)
0142 NEXT J
0144 NEXT I
0146 FNEND

END OF LISTING
```

Example of Use

```
OLD RRPADE
LIST
FILE     RRPADE


0010 DIM E(19),F(10),G(10),H(10),O(10),P(10,10)
0020 DCL S (I,J,K,K9,M9,N9,N)
0070 PRINT
0080 DISP "CODE#? 1(COEFF.S),2(DERIV.S) ";
0090 INPUT K9
0100 IF (K9-1)*(K9-2)=0 THEN 130
0110 DISP "ERROR-- ENTER 0,1(ONLY)        ↔  ";
0120 GOTO 80
0130 DISP "ENTER POLY DEGREES (BOTH <= 9) ";
0140 INPUT M9,N9
0150 IF M9*(M9-10)>=0 THEN 170
0160 IF N9*(N9-10)<0 THEN 130
0170 DISP "ERROR-- ILLEGAL VALUE          ↔";
0180 GOTO 130
0190 PRINT "TAYLOR SERIES COEFFICIENTS"
0200 FOR I=1 TO M9+N9+1 STEP 1
0210 ON K9 GOTO 220,240
0220 DISP "ENTER COEFFICIENT #";I-1;
0230 GOTO 250
0240 DISP "ENTER DERIVATIVE #";I-1;
0250 INPUT E(I)
0260 IF K9=1 THEN 320
0270 IF I<>1 THEN 300
0280 LET I9=1
0290 GOTO 320
0300 LET E(I)=E(I)/I9
0310 LET I9=I9*I
0320 PRINT E(I),
0330 NEXT I
0340 PRINT
0350 DISP "CALCULATION OF COEFFICIENT"
0360 LET Y=FNA(M9,N9)
0380 PRINT "NUMERATOR POLY DEGREE =";M9
0390 PRINT "COEFFICIENTS (IN DESCENDING ORDER)"
0400 FOR I=M9+1 TO 1 STEP -1
0410 PRINT F(I),
0420 NEXT I
0430 PRINT
0440 PRINT
0450 PRINT "DENOMINATOR POLY DEGREE =";N9
0460 PRINT "COEFFICIENTS (IN DESCENDING ORDER)"
0470 FOR I=N9+1 TO 1 STEP -1
0480 PRINT G(I),
0490 NEXT I
0500 PRINT
```

```
0510 PRINT
0520 PRINT
0530 DISP "ENTER MULTIPLIER ( > 0 )";
0540 INPUT M1
0550 IF M1>0 THEN 580
0560 DISP "ERROR--";
0570 GOTO 530
0580 IF M1=1 THEN 730
0590 PRINT "NUMERATOR COEFFICIENTS MULTIPLIED BY";M1;
0600 PRINT "(IN DESCENDING ORDER)"
0610 FOR I=N9+1 TO 1 STEP -1
0620 PRINT F(I)*M1,
0630 NEXT I
0640 PRINT
0650 PRINT
0660 PRINT "DENOMINATOR COEFFICIENTS MULTIPLIED BY";M1;
0670 PRINT "(IN DESCENDING ORDER)"
0680 FOR I=N9+1 TO 1 STEP -1
0690 PRINT G(I)*M1,
0700 NEXT I
0710 PRINT
0720 PRINT
0730 DISP "ANOTHER MULTIPL.? 1(YES),0(NO) ";
0740 INPUT M2
0750 ON M2+1 GOTO 9999,530
0760 DISP "ERROR-- ENTER 0,1(ONLY)       ??";
0770 GOTO 730

END OF LISTING



LINK *SLPADE,A
9999 END



RUN
**** FORMALLY CORRECT PROGRAM ****




CODE#? 1(COEFF.S),2(DERIV.S) ?
1
ENTER POLY DEGREES (BOTH <= 9) ?
2,3
TAYLOR SERIES COEFFICIENTS
ENTER COEFFICIENT #.0 ?
1
ENTER COEFFICIENT # 1 ?
1
ENTER COEFFICIENT # 2 ?
.5
ENTER COEFFICIENT # 3 ?
.16666667
ENTER COEFFICIENT # 4 ?
4.1666667E-02
 1                1                .5               .16666667        4.1666667E-02
ENTER COEFFICIENT # 5 ?
8.3333333E-03
 8.3333333E-03
CALCULATION OF COEFFICIENT
NUMERATOR POLY DEGREE = 2
COEFFICIENTS (IN DESCENDING ORDER)
 4.9999944E-02    .39999986        1

DENOMINATOR POLY DEGREE = 3
COEFFICIENTS (IN DESCENDING ORDER)
-1.6666683E-02    .15000008       -.60000014        1
```

```
ENTER MULTIPLIER ( > 0 )?
10
NUMERATOR COEFFICIENTS MULTIPLIED BY 10 (IN DESCENDING ORDER)
 .49999944        3.9999986        10

DENOMINATOR COEFFICIENTS MULTIPLIED BY 10 (IN DESCENDING ORDER)
-.16666683        1.5000008       -6.0000014        10

ANOTHER MULTIPL.? 1(YES),0(NO) ?
0


RUN

CODE#? 1(COEFF.S),2(DERIV.S) ?
2
ENTER POLY DEGREES (BOTH <= 9) ?
3,3
TAYLOR SERIES COEFFICIENTS
ENTER DERIVATIVE # 0 ?
1
ENTER DERIVATIVE # 1 ?
1
ENTER DERIVATIVE # 2 ?
.5
ENTER DERIVATIVE # 3 ?
1.6666667
ENTER DERIVATIVE # 4 ?
.16666667
 1                1                .25              .27777778        6.9444446E-03
ENTER DERIVATIVE # 5 ?
4.1666667E-02
ENTER DERIVATIVE # 6 ?
8.3333333E-03
 3.4722223E-04    1.1574074E-05
CALCULATION OF COEFFICIENT
NUMERATOR POLY DEGREE = 3
COEFFICIENTS (IN DESCENDING ORDER)
 .27102956        .22491776        .97556126        1

DENOMINATOR POLY DEGREE = 3
COEFFICIENTS (IN DESCENDING ORDER)
 4.9693541E-06  -6.4350400E-04  -2.4438736E-02    1


ENTER MULTIPLIER ( > 0 )?
5
NUMERATOR COEFFICIENTS MULTIPLIED BY 5 (IN DESCENDING ORDER)
 1.3551478        1.1245888        4.8778063        5

DENOMINATOR COEFFICIENTS MULTIPLIED BY 5 (IN DESCENDING ORDER)
 2.4846771E-05  -3.2175200E-03  -.12219368        5

ANOTHER MULTIPL.? 1(YES),0(NO) ?
10
ERROR-- ENTER 0,1(ONLY)        >>ANOTHER MULTIPL.? 1(YES),0(NO) ?
1
ENTER MULTIPLIER ( > 0 )?
10
NUMERATOR COEFFICIENTS MULTIPLIED BY 10 (IN DESCENDING ORDER)
 2.7102956        2.2491776        9.7556126        10

DENOMINATOR COEFFICIENTS MULTIPLIED BY 10 (IN DESCENDING ORDER)
 4.9693541E-05  -6.4350400E-03  -.24438736        10

ANOTHER MULTIPL.? 1(YES),0(NO) ?
0
```

Title — Cubic interpolation

Purpose — To determine the interpolating cubic spline for the given data set and to evaluate the cubic spline for a table of point with a fixed increment.

Method — Let (x_i,y_i) $i = 1,2,\ldots,N$ be the set of given data, where the x_i points are, in general, not evenly spaced, the aim is to find the interpolatory cubics $f_i(x)$, defined on (x_i,x_{i+1}):

$$(1)\quad f_i(x) = A_i(x-x_i)^3 + B_i\,(x-x_i)^2 + C_i\,(x-x_i) + D_i$$

where the f_i's satisfy given "smoothness" conditions.
Then it is necessary to calculate $4*(N-1)$ coefficients A_i,B_i,C_i,D_i $i = 1,2,\ldots,(N-1)$.
The function $f(x)$ on entire interval x,x_N consisting of functions $f_i(x)$ satisfying the conditions:

- exact-fit condition : $f_i(x_i) = y_i$
- continuity condition : $f_{i+1}(x_i) = y_{i+1}$
- first derivative (slope) of $f_i(x)$ and $f_{i+1}(x)$ agree at x_{i+1}
- 2nd derivative (curvature) of $f_i(x)$ and $f_{i+1}(x)$ are at x_{i+1}.
  At the end points, it is necessary to specify the second derivative only.

The used method to determine these coefficients is the Akima's method.
Having computed these coefficients it is then possible to evaluate $f(x)$ for a table of points in the interval x_A - x_N with a fixed increment.

Calling Statement — F = FNA (N,E1,E9,E)

Parameters

N = number of data points ($\leqslant 150$)
E1 = 2nd derivative at the first point
E9 = 2nd derivative at the last point
E = test for coefficients calculation
  0 = NO (already known)
  1 = YES

Global variables

Input	F( )	x-values of data (one dimension)
	G( )	y-values of data (one dimension)
	P1	start-point for table
	P2	end-point for table
	P3	increment
	(If P1 = P2 and P3 = 1 the cubic spline is evaluated point by point)	
Return	E( )	matrix of coefficients (two dimensions)
		The elements of row i (i = 2,3,...,N) are the coefficients of the function f_{i-1}
	O( )	x-values of interpolated points (one dimension)
	P( )	y-values of interpolated points (one dimension)
	P4	number of interpolated points

Status value

ø	
1	elements not in order
2	points to be interpolated outside the interval x_1 - x_N

Listing

```
LIST
FILE    *SLCSPL

0002 DEF FNA(N,E1,E9,E)X,X1,X2,Y,Y1,Y2,D1,D2,F1,F2,I
0004 DISP "CALCULATION OF COEFFICIENTS"
0006 DIM E(152,4)
0008 LET FN*=0
0010 IF E=0 THEN 140
0012 LET E(1,1)=E(1,2)=E(1,3)=0
0014 LET E(N,1)=E(N,2)=E(N,3)=0
0016 LET E(1,4)=E1
0018 LET E(N,4)=E9
0020 LET X=F(1)
0022 LET Y=G(1)
0024 LET X1=F(2)
0026 LET Y1=G(2)
0028 LET X2=F(3)
0030 LET Y2=G(3)
0032 IF X1>X THEN 38
0034 LET FN*=1
0036 GOTO 196
0038 IF X2<=X1 THEN 34
0040 LET D1=X1-X
0042 LET D2=X2-X1
0044 LET F1=Y1-Y
0046 LET F2=Y2-Y1
0048 LET E(2,4)=6*(F2/D2-F1/D1)-D1*E(1,4)
0050 LET E(2,2)=2*(X2-X)
0052 LET E(2,3)=D2
0054 FOR I=1 TO N-3 STEP 1
0056 LET X1=X2
0058 LET Y1=Y2
0060 LET D1=D2
0062 LET F1=F2
```

```
0064 LET X2=F(I+3)
0066 LET Y2=G(I+3)
0068 IF X2 =X1 THEN 34
0070 LET E(I+2,1)=D1
0072 LET E(I+2,3)=D2=X2-X1
0074 LET E(I+2,2)=2*(D2+D1)
0076 LET F2=Y2-Y1
0078 LET E(I+2,4)=6*(F2/D2-F1/D1)
0080 NEXT I
0082 LET E(2,1)=E(N-1,3)=0
0084 LET E(N-1,4)=E(N-1,4)-D2*E(N,4)
0086 LET E(2,3)=E(2,3)/E(2,2)
0088 LET E(2,4)=E(2,4)/E(2,2)
0090 LET E(2,2)=1
0092 FOR I=3 TO N-1 STEP 1
0094 LET E(I,3)=E(I,3)/(E(I,2)-E(I-1,3)*E(I,1))
0096 LET E(I,4)=(E(I,4)-E(I-1,4)*E(I,1))/(E(I,2)-E(I-1,3)*E(I,1))
0098 LET E(I,2)=1
0100 NEXT I
0102 FOR I=N-2 TO 2 STEP -1
0104 LET E(I,4)=E(I,4)-E(I,3)*E(I+1,4)
0106 NEXT I
0108 LET X1=F(1)
0110 LET Y1=G(1)
0112 LET F2=E(1,4)
0114 FOR I=2 TO N STEP 1
0116 LET X2=F(I)
0118 LET Y2=G(I)
0120 LET D1=X2-X1
0122 LET F1=Y2-Y1
0124 LET E(I,1)=(E(I,4)-F2)/(6*D1)
0126 LET E(I,2)=F2/2
0128 LET E(I,3)=F1/D1-D1*(E(I,4)+2*F2)/6
0130 LET F2=E(I,4)
0132 LET E(I,4)=Y1
0134 LET X1=X2
0136 LET Y1=Y2
0138 NEXT I
0140 DISP "INTERPOLATION"
0142 LET P4=0
0144 FOR X=P1 TO P2 STEP P3
0146 LET P4=P4+1
0148 LET X1=F(1)
0150 LET Y1=G(1)
0152 LET F2=1
0154 LET F2=F2+1
0156 LET X2=F(F2)
0158 LET Y2=G(F2)
0160 IF (X1-X)*(X-X2)>=0 THEN 172
0162 LET X1=X2
0164 LET Y1=Y2
0166 IF F2<N THEN 154
0168 LET FN*=2
0170 GOTO 196
0172 IF (X1-X)*(X-X2)<>0 THEN 188
0174 IF X<>X1 THEN 182
0176 LET Q(P4)=X4
0178 LET P(P4)=Y1
0180 GOTO 194
0182 LET Q(P4)=X2
0184 LET P(P4)=Y2
0186 GOTO 194
0188 LET Q(P4)=X
0190 LET F1=X-X1
0192 LET P(P4)=FNB(F1,F2)
0194 NEXT X
0196 FNEND
0198 DEF FNB(F1,F2)T1,I
0200 LET T1=E(F2,1)
0202 FOR I=1 TO 3 STEP 1
0204 LET T1=T1*F1+E(F2,I+1)
0206 NEXT I
0208 LET FN*=T1
0210 FNEND
```

END OF LISTING

```
OLD RRCSPL
LIST
FILE     RRCSPL


0010 FILES W""
0020 DIM F(150),G(150),D(50),P(50)
0030 SETW :1 TO 1
0040 READ :1,C,D,N
0050 IF N<=150 THEN 80
0060 PRINT "TOO MUCH DATA, ONLY FIRST 150 PT.S USED"
0070 LET N=150
0080 PRINT TAB(20),"LIST OF DATA POINTS"
0090 PRINT
0100 FOR I=1 TO N STEP 1
0110 READ :1,F(I),G(I),W
0120 PRINT F(I),G(I)
0130 NEXT I
0140 DISP "ENTER 2ND DER. AT BOTH ENDS";
0150 INPUT E1,E9
0160 DISP "INT.POINT=1,TABLE=2";
0170 INPUT R
0180 ON R GOTO 190,280
0190 DISP "ENTER X-VALUE";
0200 INPUT P1
0210 IF (C-P1)*(P1-D)>=0 THEN 250
0220 PRINT "X=";P1;"OUTSIDE THE INTERVAL"
0230 PRINT
0240 GOTO 160
0250 LET P2=P1
0260 LET P3=1
0270 GOTO 370
0280 DISP "ENTER START,END & INCR>";
0290 INPUT P1,P2,P3
0300 IF (C-P1)*(P1-D)<0 THEN 320
0310 IF (C-P2)*(P2-D)>=0 THEN 340
0320 PRINT "TABLE";P1;"-";P2;"OUTSIDE THE INTERVAL"
0330 GOTO 230
0340 IF P3>0 THEN 370
0350 PRINT "INCREMENT <=0 :";P3
0360 GOTO 230
0370 PRINT
0380 DISP "COEFF. TO CALCULATE:YES=1,NO=0";
0390 INPUT E
0400 LET Y=FNA(N,E1,E9,E)
0410 ON Y+1 GOTO 440,420,440
0420 PRINT "ERROR-ELEMENTS NOT IN ORDER"
0430 GOTO 9999
0440 IF E=0 THEN 510
0450 PRINT TAB(20);"SPLINE COEFFICIENTS (IN DESCENDING ORDER)"
0460 PRINT TAB(20);"-------------------------------------------"
0470 PRINT "  X","   A","   B","   C","   D(=Y)"
0480 FOR I=2 TO N STEP 1
0490 PRINT F(I-1),E(I,1),E(I,2),E(I,3),E(I,4)
0500 NEXT I
0510 PRINT
0520 PRINT TAB(20);"INTERPOLATION"
0530 PRINT "          X","          Y"
0540 PRINT
0550 FOR I=1 TO P4 STEP 1
0560 PRINT Q(I),P(I)
0570 NEXT I
0580 DISP "ANOTHER INTER. =1,NO=0";
0590 INPUT R1
0600 ON R1+1 GOTO 9999,160

END OF LISTING
```

```
LINK *SLCSPL,A
9999 END
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
                    LIST OF DATA POINTS

 0                  0
 1                  1
 2                  4
 3                  9
 4                  16
 5                  25
 6                  36
 7                  49
ENTER 2ND DER. AT BOTH ENDS?
0,2
INT.POINT=1,TABLE=2?
1
ENTER X-VALUE?
.5

COEFF. TO CALCULATE:YES=1,NO=0?
1
CALCULATION OF COEFFICIENTS
INTERPOLATION
                    SPLINE COEFFICIENTS (IN DESCENDING ORDER)
                    -----------------------------------------
  X                 A                 B                 C                 D(=Y)
 0                  .42264972         0                 .57735028         0
 1                 -.11324860         1.2679492         1.8452994         1
 2                  3.0344670E-02     .92820337         4.0414520         4
 3                 -8.1300813E-03     1.0192374         5.9888927         9
 4                  2.1756556E-03     .99484713         8.0029772         16
 5                 -5.7254094E-04     1.0013741         9.9991984         25
 6                  1.1450819E-04     .99965648         12.000229         36

                    INTERPOLATION
          X                  Y

 .5                 .34150636
ANOTHER INTER. =1,NO=0?
0
```

```
RUN
                    LIST OF DATA POINTS

 0                  0
 1                  1
 2                  4
 3                  9
 4                  16
 5                  25
 6                  36
 7                  49
ENTER 2ND DER. AT BOTH ENDS?
2,2
INT.POINT=1,TABLE=2?
1
ENTER X-VALUE?
.5

COEFF. TO CALCULATE:YES=1,NO=0?
1
```

```
CALCULATION OF COEFFICIENTS
INTERPOLATION
                    SPLINE COEFFICIENTS (IN DESCENDING ORDER)
                    -----------------------------------------
  X                 A                 B                 C                 D(=Y)
 0                  0                 1                 0                 0
 1                 -1.6666667E-13     1                 2                 1
 2                  1.6666667E-13     1.0000000         4                 4
 3                 -1.6666667E-13     1                 6                 9
 4                  1.6666667E-13     1.0000000         8                 16
 5                  0                 1                 10                25
 6                  0                 1                 12                36

                    INTERPOLATION
          X                 Y

 .5                .25
ANOTHER INTER. =1,NO=0?
1
INT.POINT=1,TABLE=2?
2
ENTER START,END & INCR>?
2,6,.5

COEFF. TO CALCULATE:YES=1,NO=0?
0
CALCULATION OF COEFFICIENTS
INTERPOLATION

                    INTERPOLATION
          X                 Y

 2                 4
 2.5               6.2500000
 3                 9
 3.5               12.25
 4                 16
 4.5               20.250000
 5                 25
 5.5               30.25
 6                 36
ANOTHER INTER. =1,NO=0?
0
```

References

1) Numerical Method - Robert Hornbeck - Quantum Publishers, 1975, pp. 47-50.

2) Spline Functions, Interpolation and Numerical Quadrature - T.N.E. Greville
Mathematical Methods for Digital Computers, Vol. II, John Wiley, 1967, pp. 156-168.

The P6060 Scientific Subroutine Library - Part II - Numerical Analysis - will ordinarily be supplied to you in the form of a user disk which can be found in a plastic holder inserted into the Programmer's Guide. No installation is required.
In order to be able to incorporate one or more SSL routines into a user program, be sure that a current release of the P6060 operating system is present on a system disk inserted into one drive of the floppy disk unit; then insert the SSL user disk into the other drive and follow the procedure described in the Introduction.

Obtaining a File Catalog for the SSL Library

After making certain that the machinepower is on, that the proper disk are inserted into the system, and that the message READY appears in the visual display, enter the system command CAT ⌀ U, *, , F followed by End-of-Line.
The CAT ALOG command may be entered any time the system is in Command Mode.
The result should be the listing shown below. Use this catalog listing to verify that your copy of the SSL disk is correct.

```
CAT U,*,,F

  *  R E L E A S E  1.1  *

FILE    TYPE  CREAT   LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER

SLATN2   T   081176   081176     512        512       M2200201
SLCONV   T   081176   081176     384        384       M2200201
SLRPCC   T   081176   081176     512        512       M2200201
SLPRCC   T   081176   081176     384        384       M2200201
SLCSIN   T   081176   081176     640        640       M2200201
SLCCOS   T   081176   081176     640        640       M2200201
SLCTAN   T   081176   081176     640        640       M2200201
SLCLN    T   081176   081176     512        512       M2200201
SLCTNH   T   081176   081176     640        640       M2200201
SLCCOH   T   081176   081176     640        640       M2200201
SLCSIH   T   081176   081176     640        640       M2200201
SLCZMZ   T   081176   081176     512        512       M2200201
SLCRZ    T   081176   081176     384        384       M2200201
SLPLYF   T   081176   081176     1408       1408      M2200201
SLFTRP   T   081176   081176     1280       1280      M2200201
SLLAGI   T   081176   081176     640        640       M2200201
SLFOUI   T   081176   081176     1152       1152      M2200201
SLLLSQ   T   081176   081176     3328       3328      M2200201
SLNLLS   T   081176   081176     7552       7552      M2200201
SLPADE   T   081176   081176     2304       2304      M2200201
SLFSYT   T   081176   081176     2944       2944      M2200201
SLCSQR   T   081176   081176     512        512       M2200201
SLCZDZ   T   081176   081176     512        512       M2200201
SLCZA    T   081176   081176     512        512       M2200201
SLCZN    T   081176   081176     640        640       M2200201
SLGAMA   T   151076   151076     640        640       M2200201
SLCHYP   T   081176   081176     512        512       M2200201
SLCHYF   T   081176   081176     512        512       M2200201
SLKNF    T   010976   010976     512        512       M2200201
SLCF     T   151076   151076     512        512       M2200201
SLSF     T   151076   151076     512        512       M2200201
SLEIF    T   081176   081176     512        512       M2200201
SLERF    T   081176   081176     512        512       M2200201
SLBEI    T   081176   081176     512        512       M2200201
SLBER    T   081176   081176     512        512       M2200201
SLEMF    T   010976   010976     512        512       M2200201
SLINX    T   081176   081176     640        640       M2200201
SLBES2   T   010976   010976     640        640       M2200201
SLBES1   T   010976   010976     640        640       M2200201
SLIGAM   T   151076   151076     512        512       M2200201
SLII0X   T   081176   081176     512        512       M2200201
SLFOUR   T   151076   151076     384        384       M2200201
SLPRFR   T   081176   081176     512        512       M2200201
SLPLCC   T   081176   081176     512        512       M2200201
SLPLRC   T   081176   081176     512        512       M2200201
SLPLRR   T   081176   081176     384        384       M2200201
SLLAGG   T   081176   081176     512        512       M2200201
SLHNF    T   081176   081176     512        512       M2200201
SLEINF   T   081176   081176     384        384       M2200201
SLHEN    T   081176   081176     512        512       M2200201
SLSIF    T   081176   081176     512        512       M2200201
SLCINF   T   081176   081176     512        512       M2200201
SLLEG1   T   081176   081176     512        512       M2200201
SLLEG2   T   081176   081176     512        512       M2200201
SLBJN    T   081176   081176     768        768       M2200201
SLCEXP   T   081176   081176     512        512       M2200201
OLX999   S   101276   101276     256        32        M2200201
SLCSPL   T   081176   081176     3200       3200      M2200201
```

The routines of P6060 Scientific Subroutine Library are stored as text files in the Package Library. The library is not protected against addition of new files, and the files are not secured.

The user can list each routine and, taking into account the information provided in the documentation, modify the routine when necessary to better suit the problem, and replace the standard version with the modified one.

The user can also save a different version of the routine together with the standard one, or save new routines built by himself.

For the procedure for modifying and replacing standard routines or for entering and saving new routines refer to the P6060 Reference Manual, remembering that the routines must be stored as text files and have a format of user defined function, starting with a DEF FN (letter) statement and ending with FNEND statement.

PROGRAM program-name READY TO RUN

that your program has been successfully pre-executed, by the PREPARE command.

Informational messages require no response.

Error Messages

These messages identify errors resulting from the use of P6060 commands, utility programs, or BASIC statements. The types of error they identify fall into three categories: syntax, pre-execution, and execution.

1. Syntax Errors: errors in command or BASIC statement structure (e.g., erroneous punctuation)

2. Pre-execution Errors: errors that prevent the start of execution (e.g., invalid nesting, missing END statement, etc.)

3. Execution Errors: errors detected during the execution of a program (e.g., division by zero, discrepancy between argument and operand, improper subscript values, etc.)

The system detects syntax errors as you enter each statement or command and allows you, after you press RECALL , to take immediate corrective action. The system detects pre-execution errors after you issue a PREPARE or RUN command. After notifying you of all such errors, the system switches to command mode, permitting you to make all necessary corrections. The system detects execution errors after you issue a RUN or START command or, if pre-execution has been successful, a PREPARE command. Execution errors are either recoverable or nonrecoverable.

Recoverable errors are those that can be corrected during the execution phase. When a recoverable error is detected, the system interrupts program execution, issues a warning message, and switches to debug mode. Most recoverable errors relate to invalid variable values. In these cases, the system makes an assumption for the value. To give two examples, if an attempt was made to assign the square root of a negative number to a variable, the system assumes the square root of its absolute value; if a numeric variable has not been initialized, the system assumes a value of

zero. At the time the interruption takes place, the variable is given that value. Because you are in debug mode, you have the option of changing the value assumed by the system or of accepting it. In both these cases, you can then restart execution by pressing either the STEP or CONTINUE button. You may also choose to terminate execution, by pressing the BREAK button. After BREAK is pressed, the system switches to command mode. You can then edit your program as desired.

Nonrecoverable errors are those that cannot be corrected during the execution phase. When a nonrecoverable error is detected, the ERR button lights, the system suspends program execution, issues a diagnostic message, and allows you to check the current values of the variables in your program and use calculator-mode facilities -- as you would in debug mode. However, in the case of a nonrecoverable error, you cannot use the other features of debug mode: the START command, the STEP button, or the CONTINUE button. After a nonrecoverable error occurs, you must press the BREAK button to terminate the execution of your program. (BREAK can be pressed either before or after checking the contents of the variables in your program -- but it must be pressed.) After BREAK is pressed, the system enters command mode so that the necessary corrections may be made.

A numeric code identifies each error message. In the case of pre-execution and execution errors, the code is followed by an identification of the line in which the error was made (for example, ERROR 6 IN LINE 155). The section that follows lists each code and explains the condition or conditions that caused the error. Codes 1 - 13 refer to recoverable errors detected during execution; 40 - 55 to errors that may occur during the pre-execution phase; 65 - 97 to nonrecoverable execution errors. Codes 100 - 128 refer to errors detected during the entry of a BASIC program or the compilation of a text file. Codes 151 - 156 relate to errors that may occur during an access operation to a floppy disk. Errors that may occur during the entry or execution of a system command are identified by codes 181 - 216. Codes 232 - 235 refer to utility programs and commands. The final section, abnormal termination errors, lists errors that can occur from operational malfunctions.

Error Code	Explanation
1	Either a numeric or string variable has not been initialized. The system assumes zero for a numeric variable; "null string" for a string variable.
2	The value of an argument in a built-in string function is not valid. The value returned by the function will vary according to the function specified. (See the section on built-in functions in chapter 4 for additional information.)
3	Numeric overflow. The system assumes the maximum value permitted by internal representation, with the appropriate sign.
4	Numeric underflow. The system assumes zero.
6	An attempt was made to calculate the square root of a negative number. The system assumes the square root of its absolute value.
7	A chaining operation generates a string longer than 1023 characters. The string is truncated after the first 1023 characters.
8	String overflow during the assignment of a string value to a string variable. The string is truncated at the allocation length of the variable to which it is assigned.
9	An attempt was made to calculate the logarithm of a negative number. The system assumes the logarithm of its absolute value.
10	An attempt was made to calculate the logarithm of zero. The system assumes -9.999999999999E+99.
11	An attempt was made to raise a negative number to the power of a non-integer value. The absolute value of the number is assumed and is raised to the specified power.

Recoverable errors that can occur during the execution of a BASIC program (part 1 of 2)

Error Code	Explanation
12	An attempt was made to raise zero to the power of a negative number. The system assumes +9.999999999999E+99.
13	An attempt was made to calculate the inverse of a matrix whose determinant is zero. The result of the operation is unpredictable.

Recoverable errors that can occur during the execution of a BASIC program (part 2 of 2)

Error Code	Explanation
40	A branch specified in one of the following statements is invalid: GOSUB GOTO IF...THEN MAT...READ: MAT...WRITE: ON...GOSUB ON...GOTO READ: WRITE: For complete specification information, see the explanation of the statement in error (Chapter 5).
41	NEXT not preceded by FOR or invalid nesting of two FOR/NEXT loops.
42	A multi-line function definition contains a multi-line function definition.
43	There is a reference to a function that has not been defined.
44	The maximum number of FOR/NEXT nesting levels permitted in a FOR/NEXT loop (15) has been exceeded.

Errors that can occur during the pre-execution of a BASIC program (part 1 of 2)

Error Code	Explanation
45	Use of FN* or FN*$ or FNEND outside a multi-line function definition or use of FN* within a string multi-line function definition or use of FN*$ within a numeric multi-line function definition.
46	Two nested FOR/NEXT loops use the same control variable.
47	FOR statement used with no matching NEXT.
48	A multi-line function definition lacks an FNEND statement.
49	A one- and two-dimensional array have the same name.
50	An END statement appearing in a program is not the last statement.
51	Missing END statement.
52	An attempt has been made to pre-execute a program that contains errors detected during execution of a COMPILE command, but not corrected.
53	A multi-line function definition lacks an FN* or FN*$ statement.
54	Lack of an Image statement that corresponds to a PRINT USING, DISP USING, MAT PRINT USING, or BUILD USING statement.
55	A STOP statement has been used in a multi-line function definition.

Errors that can occur during the pre-execution of a BASIC program (part 2 of 2)

Error Code	Explanation
65	No space is available in user memory to continue execution. After this error is encountered, the system switches to command mode.
66	The subscript of an array variable is invalid.
67	The operation requested would produce invalid new allocation dimensions for the specified matrix.
68	A RUN line-num or START line-num command has been used to begin execution in the middle of a FOR/NEXT loop.
69	The argument specified in a reference to a user defined function does not correspond to the type of parameter of the function.
70	RETURN statement used without GOSUB or an invalid reference has been made to a statement within a multi-line function definition.
71	An attempt has been made to assign more than 238 characters to the function keys.
72	The number of arguments specified in a reference to a user defined function does not match the number of parameters of the function.
73	The actual dimensions of a matrix do not permit the operation requested.
74	The maximum number of references to other single- or multi-line function definitions within a single- or multi-line function definition (256) has been exceeded.
75	Either matrix or string processing is requested, but the required OPTIONS command has not been entered at system initialization time.

Nonrecoverable errors that can occur during the execution of a BASIC program (part 1 of 3)

Error Code	Explanation
76	An attempt has been made to open a file which, during a preceding execution of the program, has not been closed. (To close the file, use the VALIDATE command.)
77	The file designator is either less than one or greater than the maximum number of the files that can be opened by the program at one time.
78	The operation requested for the specified file is invalid.
80	The value specified as the word number in a SETW: statement is greater than the number of words that the file can contain.
82	The requested operation is not compatible with the size of the file.
84	The EOF option has not been specified and, after the end of the file has been reached, a read operation requests additional data or a write operation attempts to continue writing.
85	The numeric expression specified as the argument of a TAB function has been evaluated as less than 1.
86	An attempt has been made to assign a string value to a numeric variable.
87	In a BBUILD statement, the allocation length of the specified string variable is not sufficient to allow the assignement of all the data resulting from the evaluation of its expressions.
88	Either a READ statement has requested additional data and the program's internal file contains no more data or, for an ASSIGN statement, the number of data items resulting from the evaluation of the string expression is less than the number of variables to which they must be assigned.

Nonrecoverable errors that can occur during the execution of a BASIC program (part 2 of 3)

Error Code	Explanation
89	The image field is invalid for data specified in a BUILD USING, DISP USING, MAT PRINT USING, or PRINT USING statement.
90	An attempt has been made to convert a value greater than 255 or less than 0 into an ISO character.
91	In a CONVERT statement, the numeric expression assigned as the value of the LENGTH operand has been evaluated as negative.
92	Invalid file name specified in a CHAIN statement.
93	In a BASSIGN, MAT READ:, or READ: statement, an attempt has been made to assign a string value to a numeric variable or vice versa.
96	The value specified as the word number in a SETW: statement is less than or equal to zero.
97	A SCRATCH: or APPEND: statement refers to a random file.

Nonrecoverable errors that can occur during the execution of a BASIC program (part 3 of 3)

Error Code	Explanation
100	Only a line number has been specified.
101	Invalid line number.
102	Invalid keyword.
103	Invalid operand.
104	Invalid expression.
105	Type discrepancy between operand and operator.
106	The arguments specified in a reference to a function are wrong either in number or type.
107	Invalid file name.
109	Non-interpretable syntax error.
110	The function being defined has already been defined in another DEF statement.
111	An attempt has been made to cross-reference more than 255 lines.
112	The number of numeric or string variables previously referred to in the program is the maximum permitted.
113	Invalid character. (This error may occur in the case of unbalanced parentheses.)
114	Recursive definition in a single-line user-defined function.
115	Invalid reference to a variable or function.
117	No space is available in user memory to accept the keyboard entry.
118	The program already contains a FILES statement.

Errors that can occur when entering a program or compiling a text file or in calculator mode (part 1 of 2)

Error Code	Explanation
119	The number of functions that can be defined or re-defined in a program is currently at its maximum.
120	The line number referred to does not exist in the program.
128	Too many operations have been attempted in a single statement.

Errors that can occur when entering a program or compiling a text file or in calculator mode (part 2 of 2)

Error Code	Explanation
151	Operational problem on floppy disk drive 1 (upper drive).
152	Operational problem on floppy disk drive 2 (lower drive).
156	There is no system floppy disk in the unit.

Errors that can occur in access to a floppy disk

Error Code	Explanation
181	Insufficient memory to execute the requested operation.
182	The line number option (#) specified in a TRANSCODE statement is invalid for the requested operation.
183	No space has been allocated for the specified library.
184	The user floppy disk has not been initialized or reference has been made to a user floppy disk when none is in the drive.
185	The system floppy disk has not been initialized to contain an application library.

Errors that can occur during the entry or execution of a system command (part 1 of 3)

Error Code	Explanation
186	The specified file name duplicates the name of an existing file.
187	A specified file cannot be found.
188	Insufficient space available on the floppy disk or in the specified library for the requested operation.
189	Invalid attempt to decrease the size of a file.
190	The command is not recognized.
191	No file name specified.
192	Invalid character specified.
193	A required operand has not been specified.
194	Specified line number cannot be found.
195	An attempt has been made to use the START command for a program that was previously stored without pre-execution.
196	Invalid operand.
197	The line number specified in a START command is part of a multi-line function definition.
198	The space requested exceeds the space available.
199	The requested operation is not accepted for a protected program.
200	The requested operation is not accepted for a protected library.
201	The requested operation requires a double floppy disk unit.

Errors that can occur during the entry or execution of a system command (part 2 of 3)

Error Code	Explanation
202	The requested operation is valid only for systems having a printer.
203	The first line number specified is greater than the second line number.
205	The requested operation is invalid for a protected line.
206	The file present in main memory is not a program.
207	The requested operation is invalid for the file type.
208	The option specified is not available with the system.
209	A line number greater than 9999 has been generated.
210	The X option is invalid for a program.
211	There is no program or file in main memory.
212	The line or lines to be printed do not exist.
213	The length of the line prevents its listing, display, or the compilation.
214	Attempt to link a multi-line function definition that has no DEF statement.
216	A program for which the compilation has been specified contains a branch to a line number that does not exist.

Errors that can occur during the entry or execution of a system command (part 3 of 3)

Error Code	Explanation
232	The sum of $n_1 + n_2 + n_3$ is greater than 14.
234	The name of the utility program has been omitted.
235	Invalid utility program name.

Errors that can occur during the calling or execution of a utility program

Error Code	Explanation
4A *	Main memory is damaged; its contents has been deleted.
12A * 16A *	The system floppy disk is damaged; the contents of the disk are invalid. The contents of main memory are deleted.
ABN FD *	The upper drive of the floppy disk unit is not working properly. (Check if the flap is closed.)
ABN FD**	The lower drive of the floppy disk unit is not working properly. (Check if the flap is closed.)
ABN PRT	The integrated printer is not working properly. (Check the position of the release lever.)

Abnormal termination errors

Note: Other error codes similar in form to those listed above may be issued when the system encounters an abnormal operational condition. In the case of such errors, and of the ones above, pressing the [redacted] button can sometimes correct the error condition. If you press [redacted] and the READY message appears, retry the operation that resulted in the error. If READY fails to appear, try switching off the power, waiting a few seconds, and switching the power back on. If READY does not appear, contact your nearest Olivetti technical representative.