

M20

OLINUM
(Numerical Analysis)

User Guide

olivetti L1

50

50

50

50

50

M20

OLINUM
(Numerical Analysis)

User Guide

olivetti L1

PREFACE

This document contains a set of subroutines that provides solutions for problems in numeric analysis. These subroutines are intended for the applications of Users who provide their own programming. The use of these subroutines permits a more rapid analysis and development of the applications.

RELATED PUBLICATIONS: None

DISTRIBUTION: General (G)

FIRST EDITION: July 1982

MULTIPLAN is a registered trademark
of the MICROSOFT INC.

PUBLICATION ISSUED BY:

Ing. C. Olivetti S.p.A
Servizio Centrale Documentazione
77, Via Jervis-10015 IVREA (Italy)

© 1982, by Olivetti

The following are trademarks of Ing. C. Olivetti S.p.A.: OLICOM, OLITERM, OLIWORD, OLINUM, OLISTAT, OLIMASTER, OLITUTOR, OLITEST, OLIENTRY, OLISORT, GTL.

5

5

5

5

5

5

The following are trademarks of Ing. C. Olivetti S.p.A.: OLICOM, OLITERM,
OLIWORD, OLINUM, OLISTAT, OLIMASTER, OLITUTOR, OLITEST, OLIENTRY,
OLISORT, GTL.

OPERATIONS LIBRARY

MULTIPLAN
USER GUIDE

3983370 D

OLIWORD
USER GUIDE (*)

3983390 P

OLIENTRY
USER GUIDE (*)

3983410 R

OLIMASTER
USER GUIDE

3983430 K

OLISTAT (Statistical Analysis)
USER GUIDE

3983450 M

OLINC
USER GUIDE (*)

3983470 P

OLISORT
USER GUIDE

3982260 E

OLINUM (Numerical Analysis)
USER GUIDE

3987570 J

INSTALLATION
MANUAL

3966250 U

PRINTER PR 2400
OPERATING GUIDE

3984150 U (1)

PRINTER PR 1450
OPERATING GUIDE

3984170 W (1)

PRINTER PR 1471
OPERATING GUIDE

3984190 G (1)

PRINTER PR 1481
OPERATING GUIDE (*)

3984240 V

PRINTER PR 430
OPERATING GUIDE (*)

3982940 B

INTRODUCTION TO
THE SYSTEM

3930110 M (1)

APPLICATION SOFTWARE LIBRARY

OLITUTOR
USER GUIDE

3983350 B

PCOS (Professional Computer
Operating System)
USER GUIDE

3987590 U

BASIC LANGUAGE
REFERENCE GUIDE

3982430 P (1)

BASIC & PCOS
POCKET REFERENCE

3987640 R

ASSEMBLER LANGUAGE
REFERENCE GUIDE

3982410 M

ASSEMBLER LANGUAGE
USER GUIDE

3987670 L

ASSEMBLER LANGUAGE
POCKET REFERENCE (*)

3983710 P

ISAM (Index Sequential
Access Method)
USER GUIDE

3987690 W

FORTRAN
REFERENCE GUIDE (*)

3987730 J

PASCAL
REFERENCE GUIDE (*)

3987710 Q

PROGRAMMING LIBRARY

OLITEST
USER GUIDE

3982330 D

I/O WITH EXTERNAL
PERIPHERALS
USER GUIDE

3982300 N (1)

VIDEOTEX
USER GUIDE (*)

3982280 Q

OLICOM
USER GUIDE

3983650 R

OLITERM
USER GUIDE

3983690 V

ADVANCED FACILITIES LIBRARY



CONTENTS

1. INTRODUCTION

WHAT IS OLINUM 1-1

HOW TO USE OLINUM 1-1

INPUT TO OLINUM SUBROUTINES 1-1

OUTPUT FROM OLINUM SUBROUTINES 1-2

HOW THIS MANUAL IS ORGANISED 1-2

2. HOW TO BEGIN

INTRODUCTION 2-1

POWER ON 2-1

STARTING THE APPLICATION PACKAGE 2-4

USING THE TEST PROGRAMS 2-5

3. COMBINATORIAL ANALYSIS 3-1

SLCONN. Number System Conversion 3-1

SLPRIM. Factoring and Prime Number Generation 3-4

SLEUCL. Greatest Common Denominator and Lowest Common Multiple 3-7

SLRFCO. Rational Fraction Conversion to Continued Fraction 3-9

SLSUCO. Quadratic Surd Conversion to Continued Fraction 3-11

SLCFCO. Convergents of a Continued Fraction 3-13

SLFACT. Factorial and Log Factorial 3-16

SLBINO. Binomial Coefficients 3-19

SLMULT. Multinomial Coefficients 3-21

SLDUPL. Probability of Duplication in a Given Universe 3-23

4. ELEMENTARY FUNCTIONS

SLATN2. Arctangent of a Ratio 4-1

SLCONV. Reduction of an Angle to the First Quadrant 4-3

SLRPCC. Rectangular to Polar Coordinates Conversion 4-5

SLPRCC. Polar to Rectangular Coordinates Conversion 4-7

5. ELEMENTARY FUNCTIONS (COMPLEX)

SLCSIN. Sine of a Complex Number 5-1

SLCCOS.	Cosine of a Complex Number	5-3
SLCTAN.	Tangent of a Complex Number	5-5
SLCASN.	Arcsine of a Complex Number	5-7
SLCACs.	Arcosine of a Complex Number	5-9
SLCATN.	Arctangent of a Complex Number	5-11
SLCLN.	Natural Logarithm of a Complex Number	5-13
SLCEXP.	Exponential of a Complex Number	5-15
SLCRZ.	Reciprocal of a Complex Number	5-17
SLCZMZ.	Multiplication of Two Complex Numbers	5-19
SLCZDZ.	Division of Two Complex Numbers	5-22
SLCSQR.	Square Root of a Complex Number	5-25
SLCZN.	Integral Power of a Complex Number (Z^n Recurrence)	5-27
SLCZA.	Real Power of a Complex Number	5-29

6. POLYNOMIALS

SLPLRC.	Evaluation of Real Polynomials (Complex Argument)	6-1
SLPLRR.	Evaluation of Real Polynomials (Real Argument)	6-3
SLPRRR.	Calculating the Coefficients of a Polynomial from Roots	6-5
SLPLYM.	Multiplication of Two Real Polynomials	6-7
SLPLYD.	Division of Two Real Polynomials	6-9
SLPTRA.	Translation of Coefficients of a Real Polynomial	6-12

7. HIGHER MATHEMATICAL FUNCTIONS

SLKMF.	Complete Elliptic Integral of First Kind	7-1
SLEMF.	Complete Elliptic Integral of Second Kind	7-3
SLLAGG.	Generalised Laguerre Polynomial $Ln^{(a)}(x)$	7-5
SLHNF.	Hermite Polynomial $Hn^{(x)}$	7-7

CONTENTS

SLHEN.	Hermite Polynomial $He_n(x)$	7-9
SLFOUR.	Evaluation of Fourier Series	7-11
SLGAMA.	Gamma Function	7-13
SLERF.	Error Function $\text{erf}(x)$	7-15
SLBJN.	Bessel Function of Integer Order $J_n(x)$	7-17
SLIIØX.	Definite Integral of the Bessel Function $I_0(x)$	7-19
SLSF.	Fresnel Integral $S(x)$	7-21
SLCF.	Fresnel Integral $C(x)$	7-23
SLCHYF.	Confluent Hypergeometric Function	7-25
SLGHYP.	Gauss Hypergeometric Function	7-27
SLSIF.	Sine Integral $\text{Si}(x)$	7-29
SLCINF.	Cosine Integral $\text{Ci}(x)$	7-31
SLEIF.	Exponential Integral $\text{Ei}(x)$	7-33
SLEINF.	Exponential Integral $\text{Ein}(x)$	7-35

8. SOLUTION OF EQUATIONS

SLBAIR.	Roots of a Real Polynomial (Newton-Bairstow)	8-1
SLRBIS.	Roots of a Real Function (Bisection)	8-4
SLNLIN.	Solution of a Non-linear System	8-7

9. LINEAR ALGEBRA

SLCR0.	Solution of a Linear System and Matrix Inversion for a General Matrix	9-1
SLCH0.	Solution of a Linear System and Matrix Inversion for a Symmetric Positive-definite Matrix	9-5
SLJAC.	Eigenvalues and Eigenvectors of a Symmetric Matrix using the Jacobi Method	9-10
SLHES.	Eigenvalues of a General Matrix using the Q-R Algorithm	9-13
SLHEV.	Eigenvectors of a General Matrix	9-17

SLCHA.	Characteristic Polynomial of a General Matrix	9-19	SLPADE.	Rational Function Fitting (Padè Approximation)	10-28
SLJCB.	Evaluation of the Jacobian Matrix	9-23	11. NUMERICAL INTEGRATION AND DIFFERENTIATION		
10. CURVE FITTING AND INTERPOLATION			SLITEQ.	Integration of a Tabulated Function (Equal Spacing)	11-1
SLLSQ.	Least Squares Curve Fitting to User Supplied Basis	10-1	SLDTEQ.	Differentiation of a Tabulated Function (Equal Spacing)	11-5
SLNLLS.	Non-linear Least Squares Curve Fitting to an Arbitrary Scalar Function	10-4	SLDIFF.	Differentiation of a Non-tabulated Function	11-9
SLFSYT.	Weighted Least Squares Orthogonal Polynomial Curve Fit (Forsythe)	10-8	SLROMB.	Romberg Integration	11-13
SLFTRP.	Coefficients of Fourier Series to Represent Discrete Data	10-12	SLGAUS.	Guass-Legendre Quadrature	11-16
SLFOUI.	Fourier Interpolation	10-17	SLLAGU.	Guass-Laguerre Quadrature	11-19
SLPLYF.	Lagrangian Interpolation	10-19	12. ORDINARY DIFFERENTIAL EQUATIONS		
SLCSPC.	Cubic Spline	10-23	SLEROM.	Integration of Differential Equations (Euler-Romberg Method)	12-1
SLCSPI.	Cubic Spline Interpolation	10-24	A. CALCULATION METHODS		
			<u>COMBINATORIAL ANALYSIS</u>		
			SLCONN.		A-1

CONTENTS

SLPRIM.	A-2	SLCZDZ.	A-10
SLEUCL.	A-2	SLCSQR.	A-10
SLRFCO.	A-3	SLCZN.	A-10
SLSUCO.	A-3	SLCZA.	A-11
SLCFCO.	A-4	<u>POLYNOMIALS</u>	A-11
SLFACT.	A-4	SLPLRC.	A-11
SLBINO.	A-5	SLPLRR.	A-11
SLMULT.	A-5	SLPRRR.	A-11
SLDUPL.	A-6	SLPLYM.	A-12
<u>ELEMENTARY FUNCTIONS</u>	A-6	<u>HIGHER MATHEMATICAL FUNCTIONS</u>	A-12
SLRPCC.	A-6	SLKMF.	A-12
SLPRCC.	A-6	SLEMF.	A-12
<u>ELEMENTARY FUNCTIONS (COMPLEX)</u>	A-7	SLLAGG.	A-12
SLCSIN.	A-7	SLHNF.	A-13
SLCOS.	A-7	SLHEN.	A-13
SLCTAN.	A-8	SLFOUR.	A-13
SLCASN.	A-8	SLGAMA.	A-13
SLCACS.	A-8	SLERF.	A-13
SLCATN.	A-8	SLBJN.	A-14
SLCLN.	A-9	SLIIØX.	A-14
SLCEXP.	A-9	SLSF.	A-14
SLCRZ.	A-9	SLCF.	A-14
SLCZMZ.	A-9	SLCHYF.	A-15

SLGHYP.	A-15	SLPLYF.	A-29
SLSIF.	A-15	SLCSPC.	A-31
SLCINF.	A-15	SLCSPI.	A-32
SLEIF.	A-15	SLPADE.	A-32
SLEINF.	A-16	<u>NUMERICAL INTEGRATION AND DIFFERENTIATION</u>	A-34
<u>SOLUTION OF EQUATIONS</u>	A-16	SLITEQ.	A-34
SLBAIR.	A-16	SLDTEQ.	A-35
SLRBIS	A-16	SLDIFF.	A-36
SLNLIN.	A-17	SLROMB.	A-36
<u>LINEAR ALGEBRA</u>	A-18	SLGAUS.	A-37
SLCRO.	A-18	SLLAGU.	A-38
SLCHO.	A-18	<u>ORDINARY DIFFERENTIAL EQUATIONS</u>	A-38
SLJAC.	A-18	SLEROM.	A-38
SLHES.	A-19		
SLHEV.	A-19	B. SUBROUTINES AND RE- LATED TEST PROGRAMS	
SLCHA.	A-20		
SLJCB.	A-20	SUBROUTINES AND RE- LATED TEST PROGRAMS	B-1
<u>CURVE FITTING AND INTERPOLATION</u>	A-20		
SLLLSQ.	A-20		
SLNLLS.	A-22		
SLFSYT.	A-25		
SLFTRP.	A-26		
SLFOUI.	A-28		

1. INTRODUCTION

ABOUT THIS CHAPTER

This chapter provides general information about OLINUM and a brief explanation of how this manual is organised.

CONTENTS

<u>WHAT IS OLINUM</u>	1-1
<u>HOW TO USE OLINUM</u>	1-1
INPUT TO OLINUM SUBROUTINES	1-1
OUTPUT FROM OLINUM SUB- ROUTINES	1-2
<u>HOW THIS MANUAL IS ORGANISED</u>	1-2

INTRODUCTION

WHAT IS OLINUM

The M20 Scientific Subroutine Library OLINUM provides users who develop their own application programs with a set of pre-written mathematical routines which can easily be incorporated into their work, forming single program units. The routines described in this volume have been designed to solve the more common computational problems in mathematics. The routines are free of input/output; this enables one or more routines to be used in the sequence appropriate to the computational requirements of the particular application program.

The routines are written in BASIC and take the form of subroutines. They are stored on a user floppy disk; each routine is in a single file with a filename composed of the characters sl followed by a mnemonic related to the computational method and completed with a full stop. The routines do not include input (or output) statements but operate on data already in memory; there is no fixed maximum dimensioning of the data array on which the routines operate.

HOW TO USE OLINUM

In order to build a BASIC program incorporating one or more routines from OLINUM, the user must write the main program taking into account the following points:

- the program must contain input and output statements which use the same global variables as the routines
- the program must contain the necessary DIM and DCL statements for the global arrays
- the OLINUM subroutine is referred in the calling statement GOSUB 'address'; the address being fixed for each specified subroutine
- if several subroutines are incorporated in the same program each of them must be called by it's corresponding GOSUB statement
- the program must not contain any line number which is used for the subroutines

INPUT TO OLINUM SUBROUTINES

Before calling the routine the user must store the data on which the

routine is to operate, taking into account that data may be in the form of single numeric variables or numerical arrays of one or two dimensions.

In the case of single numeric variables, they can be used as function parameters and the user must select the address of the function and provide the parameters for the function to be called before the statement for calling it (GOSUB). In the case of numeric arrays they are global variables for the program and the subroutine and must be dimensioned in the program, declared in single or double precision as required and have the same name in the function.

The name of the parameters is associated with the name of the subroutine as well as the 'local' variables of which it makes use, so that there will be no confusion with those used for calling the program.

OUTPUT FROM OLINUM SUBROUTINES

The results of a routine may be a single numeric variable (which may be given by the return value FUN. of the function) or more than one numeric variable or array; in the latter case the results will be global variables and must be dimensioned and declared in the program.

Care must be exercised in choosing the names of application program variables. If the same names are used both for application program variables and subroutine internal variables, the value assigned to the variables in the application program will be changed after the subroutine calling statement.

HOW THIS MANUAL IS ORGANISED

The second chapter of this manual; How to Begin, describes how to load the System and Application disks on the M20 and how to incorporate the OLINUM subroutines. The OLINUM subroutines are then described in 9 separate chapters, each chapter corresponding to a main area of Numerical analysis. Within each chapter each subroutine is described in tabular form as follows:

First Table (see Fig 1-1):

- The subroutine title, i.e. SL....
- A brief description of the purpose of the subroutine.

INTRODUCTION

- The input parameters listed with their meanings plus their values, where known.
- The output parameters listed with their meanings plus their values, where known. This section includes the possible FUN. values returned by the subroutine; in general FUN. = 0 signifies a correct calculation and FUN. = an integer greater than 0 signifies an error.
- The program variables or arrays required for calculation, if any.
- The calling statement GOSUB
- A cross-reference to the Appendix which contains the method used to calculate the subroutine
- The calling statements of any associated subroutines.

This first table is followed by any general notes which are of importance to the correct utilisation of the subroutine.

SUBROUTINE TITLE

INPUT		
VARIABLE	DESCRIPTION	VALUE
Parameter1 Parameter2 Parameter _n		
OUTPUT		
Parameter1 Parameter2 Parameter _n		
CALLING SEQUENCE GOSUB ...		
METHOD See Appendix ...		
CALLED SUBROUTINES		

Figure 1-1 Subroutine Table

INTRODUCTION

The second table describes the Test Program for a subroutine and is laid-out as follows (see Fig 1-2)

- The subroutine title, i.e. SL
- The Test Program name
- Any notes considered relevant to the Test Program
- The subroutine title or titles required for the Merge
- An input and an output section which show the actual data input and output during the Test Program

The Manual is concluded by two Appendices; the first containing a description of the Methods used for each subroutine and the second containing a list of the subroutines with their related Test Programs

TEST PROGRAM FOR SUBROUTINE TITLE

TEST PROGRAM NAME:	
Note: Any relevant notes	
MERGE: Subroutine title(s)	
INPUT	OUTPUT
Data input by user	Data output by machine

Figure 1-2 Test Program Table

2. HOW TO BEGIN

ABOUT THIS CHAPTER

This chapter contains an explanation of how to start the Olinum application package on the M20 Computer System.

CONTENTS

<u>INTRODUCTION</u>	2-1
<u>POWER ON</u>	2-1
<u>STARTING THE APPLICATION PACKAGE</u>	2-4
<u>USING THE TEST PROGRAMS</u>	2-5

HOW TO BEGIN

INTRODUCTION

The instructions which follow are given in brief. For a complete discussion of the M20 from an operational point of view refer to the Professional Computer Operating System PCOS User Guide code 3930090 C

POWER ON

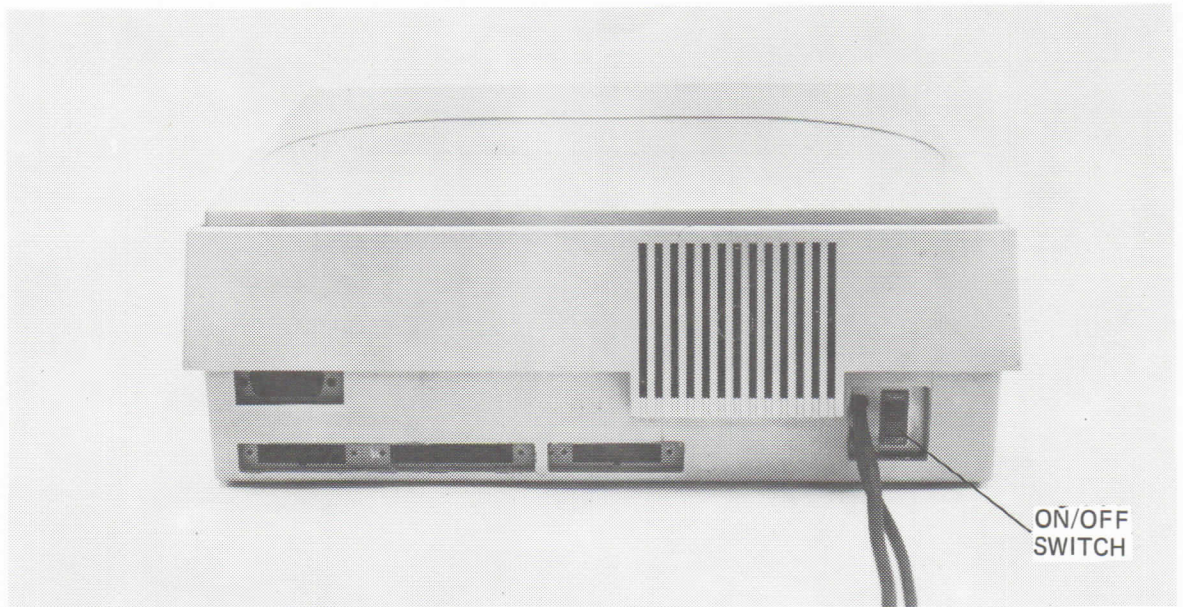
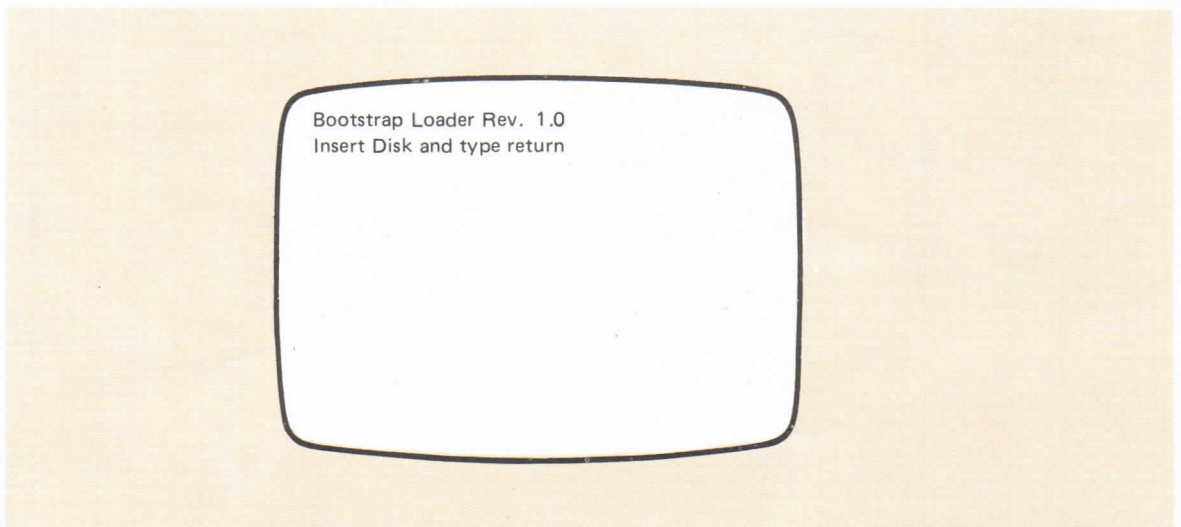


Figure 2-1 ON/OFF SWITCH

- 1) Press the ON/OFF switch on the rear of the machine to the ON position. The red power-on-light at the top left of the keyboard, comes on and the following appears on the screen.



2. Insert the System Disk (Figure 2-2) into the right hand disk drive as indicated in Figure 2-3. Press the **CR** key.

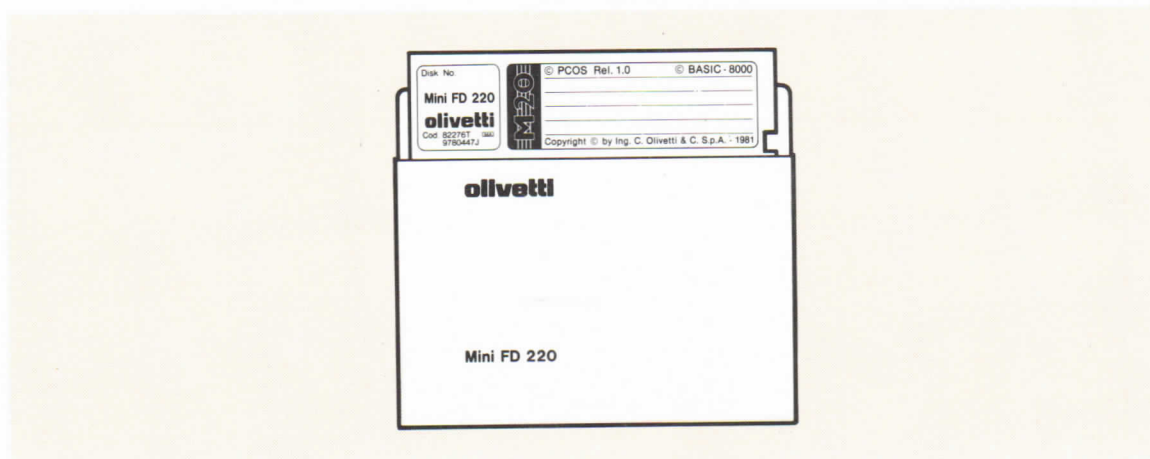
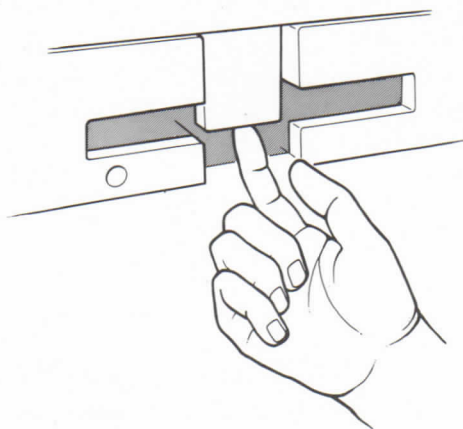
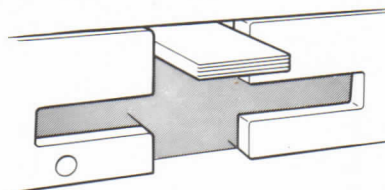


Figure 2-2

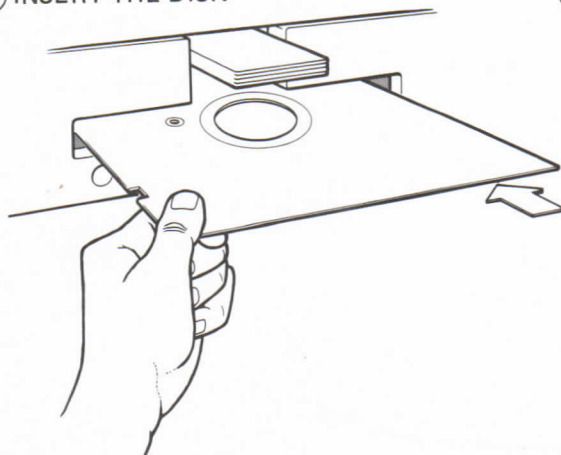
- ① PULL THE COVER OUTWARDS



- ② LET IT FLAP OPEN



- ③ INSERT THE DISK



- ④ PRESS THE DRIVE COVER DOWN AND LET IT FLAP CLOSED

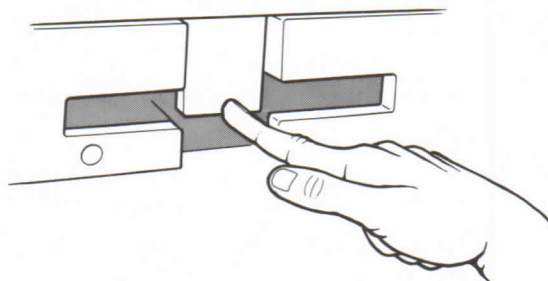


Figure 2-3 Loading a Disk

HOW TO BEGIN

The screen illustrated in Figure 2-4 appears:

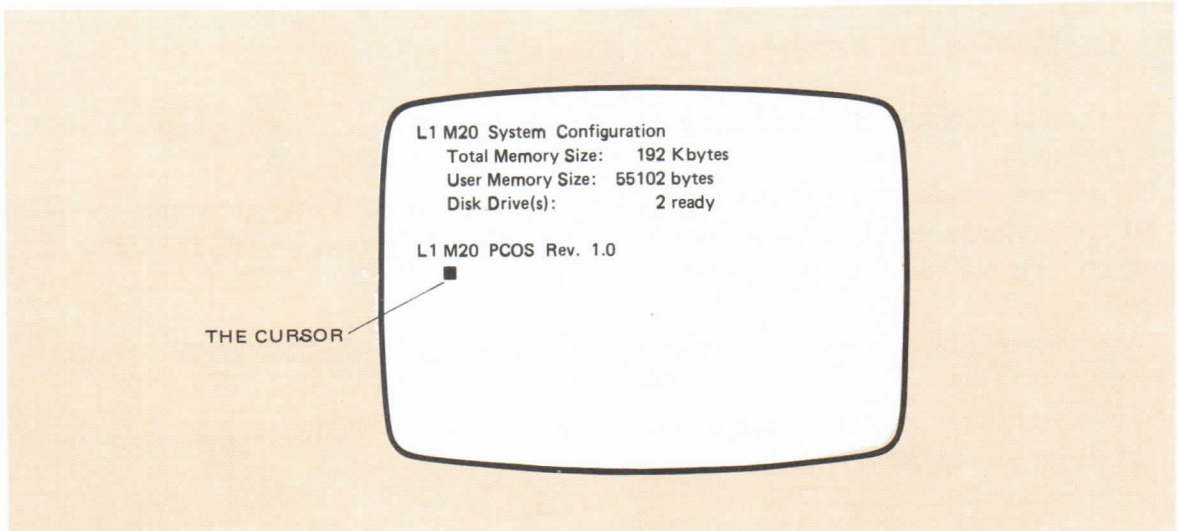


Figure 2-4 Start-up Screen

The numbers which appear on your screen may be slightly different from those shown in Figure 2-4. This depends on the configuration you are using and will not affect the running of your application. If the start-up procedure does not go as planned you should refer to the "PCOS User Guide" code number 3930090 C, Chapter 8 "TROUBLESHOOTING".

STARTING THE APPLICATION PACKAGE

When the System disk has been loaded as described in the previous section the Application can be started as follows:

- Enter the Command New and Load the main program
- Enter the merge command for the first called subroutine with the format; Merge <filename>, where filename is the name of the file in which the subroutine is stored (i.e. SL)
- Enter a Merge Command for each of the other subroutines called, if any
- The program plus the Merged Subroutines are now ready to run and maybe listed or saved as any other program.

Example

For this example the test program for the subroutine SLCONN. will be the application program; and a listing is provided to show the point of merging:

```
Load "1: SLCONN
O.K.
MERGE "1: SLCONN.
O.K.
RUN.
```

```
10 OPTION BASE 1 : DEFDBL A-Z : PI=3.14159265359#
20 INPUT "ENTER BASE1,BASE2  ",B1,B2
30 INPUT "ENTER INTEGER PART  ",N
40 A$="Base "+STR$(B1)
50 PRINT
60 PRINT USING "\      \Integer Part  ##,#####^####";A$,N
70 INPUT "ENTER FRACTIONAL PART  ",F
80 PRINT USING "      Fractional Part ##,#####^####";F
90 PRINT
100 PCONN1=N:PCONN2=F:PCONN3=B1:PCONN4=B2:PCONN5=C1
110 GOSUB 30401:FUNCONN=FUN.
120 IF ((FUNCONN+1)<.5) OR ((FUNCONN+1) 6 ) GOTO 140
130 ON FUNCONN+1 GOTO 140,200,220,240,260,280
140 A$="Base "+STR$(B2)
150 PRINT USING "\      \Integer Part  ##,#####^####";A$,R
160 PRINT USING "      Fractional Part ##,#####^####";R2
170 PRINT
180 PRINT
190 GOTO 20
200 PRINT "INTEGER PART AN INTEGER )=0 ONLY"
210 GOTO 30
```


HOW TO BEGIN

```
220 PRINT "0<= FRACTIONAL PART<1 ONLY"  
230 GOTO 70  
240 PRINT "BASE MUST BE AN INTEGER >1"  
250 GOTO 20  
260 PRINT "ERROR IN INTEGER DIGITS"  
270 GOTO 30  
280 PRINT "ERROR IN FRACTIONAL DIGITS"  
290 GOTO 70  
30401 PI=3.14159265359#  
30402 REM SUBROUTINE
```

It is also possible to load the subroutines and then specify the line number for calling a particular subroutine in the application program.

USING THE TEST PROGRAMS

To test the subroutines the OLINUM disk should be loaded as described previously and "ba" entered via the keyboard. If a printer is available and a print-out required then "ba + prt" must be entered via the keyboard.

The test program is called by entering;

load "1: test program name

and the subroutine is merged with the test program by entering

merge "1: subroutine name.

If the merge is successful then O.K. will appear on the screen and the test program can be started by entering RUN. The input for the various test programs is described in the tables related to the subroutines.

To cancel a test program the blue **SHIFT** key and the **C** key must be depressed simultaneously.

50

50

50

50

50

3. COMBINATORIAL ANALYSIS

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of problems in Combinatorial Analysis.

CONTENTS

SLCONN.	Number System Conversion	3-1	SLMULT.	Multinomial Coefficients	3-21
SLPRIM.	Factoring and Prime Number Generation	3-4	SLDUPL.	Probability of Duplication in a Given Universe	3-23
SLEUCL.	Greatest Common Denominator and Lowest Common Multiple	3-7			
SLRFCO.	Rational Fraction Conversion to Continued Fraction	3-9			
SLSUCO.	Quadratic Surd Conversion to Continued Fraction	3-11			
SLCFCO.	Convergents of a Continued Fraction	3-13			
SLFACT.	Factorial and Log Factorial	3-16			
SLBINO.	Binomial Coefficients	3-19			

NUMBER SYSTEM CONVERSION

To convert an integer or decimal fraction from one polynomial base to another

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCONN1 PCONN2 PCONN3 PCONN4	Integer part of number decimal fraction part of number conversion from base PCONN3 to base PCONN4	PCONN3, PCONN4 integers >1
OUTPUT		
R R2 FUN.	integer part of converted number decimal fraction of converted number Return Status	$\emptyset$ = correct calculation 1 = PCONN1 other than an integer $\geq \emptyset$ 2 = PCONN2 not in the range $\emptyset \leq \text{PCONN1} \leq 1$ 3 = PCONN3 and/or PCONN4 other than integers > 1 4 = digit of PCONN1 $\geq$ PCONN3 5 = digit of PCONN2 $\geq$ PCONN3

CALLING SEQUENCE GOSUB 30401
METHOD See Appendix A
CALLED SUBROUTINES

COMBINATORIAL ANALYSIS

TEST PROGRAM FOR SLCONN.

TEST PROGRAM NAME: SLCONN	
Note:	
MERGE: SLCONN.	
INPUT	OUTPUT
ENTER BASE1, BASE2 10,2 ENTER INTEGER PART 123 ENTER FRACTIONAL PART .258 ENTER BASE1,BASE2 16,10 ENTER INTEGER PART 10605 ENTER FRACTIONAL PART .208070211 ENTER FRACTIONAL PART .0208070211 ENTER BASE1,BASE2	BASE 10, INTEGER PART 1.23000000 000000D+002 FRACTIONAL PART 2.58000000000000 -001 BASE2, INTEGER PART 1.1110110000000+ 006 FRACTIONAL PART 1.0000100000011D -002 BASE 16 , INTEGER PART 1.06050000 0000D+004 FRACTIONAL PART 2.0807021100000 -001 ERROR IN FRACTIONAL DIGITS FRACTIONAL PART 2.0807021100000 -002 BASE 10, INTEGER PART 3.57000000 0000D+002 FRACTIONAL PART 1.579996386963D -001

FACTORING AND PRIME NUMBER GENERATION

To determine whether or not a given integer is prime, or to decompose a given integer into its prime factors

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPRIM1 PPRIM2	positive integer determination or decomposition of PPRIM1	1 = determine whether or not PPRIM1 is prime 2 = decompose PPRIM1 into its prime factors
OUTPUT		
I1 A(I1,2) FUN.	number of prime factors if PPRIM2 = 2 two dimensional array containing each prime factor, A(J,1) and its corresponding exponent, A(J,2). None if PPRIM2 = 1 Return Status	J = 1,2...,I1 if PPRIM2 = 2 Ø = PPRIM1 is prime 1 = PPRIM1 is not prime 2 = PPRIM1 is not a positive integer 3 = PPRIM2 is other than 1 or 2

COMBINATORIAL ANALYSIS

CALLING SEQUENCE GOSUB 30601
METHOD See Appendix A
CALLED SUBROUTINES

TEST PROGRAM FOR SLPRIM.

TEST PROGRAM NAME: SLPRIM	
Note:	
MERGE: SLPRIM.	
INPUT	OUTPUT
ENTER1 (PRIMES) OR 2(FACTORS)1 ENTER LIMITS 1,15 ENTER1 (PRIMES) OR 2(FACTORS) 1 ENTER LIMITS 23.5, 26 ENTER1 (PRIMES) OR 2(FACTORS) 2 ENTER INTEGER 180 ENTER1 (PRIMES) OR 2(FACTORS) 2 ENTER INTEGER 37 ENTER1 (PRIMES) OR 2(FACTORS)	PRIME NUMBERS BETWEEN 1 AND 15 1 2 3 5 7 11 13 PRIME NUMBERS BETWEEN 23.5 AND 26 POSITIVE INTEGERS ONLY PRIME FACTORS OF 180 $180 = 2^2 * 3^2 * 5$ PRIME FACTORS OF 37 37 IS PRIME

COMBINATORIAL ANALYSIS

GREATEST COMMON DENOMINATOR AND LOWEST COMMON MULTIPLE

SLEUCL.

To determine the Greatest Common Denominator (G.C.D) and Lowest Common Multiple (L.C.M) of two positive integers N1 and N2.

INPUT		
VARIABLE	DESCRIPTION	VALUE
PEUCL1 PEUCL2	positive integers whose GCD and LCD are to be found	
OUTPUT		
G L FUN.	GCD of PEUCL1 and PEUCL2 LCD of PEUCL1 and PEUCL2 Return Status	 Ø = correct calculation 1 = PEUCL1 and/or PEUCL2 not positive integers
CALLING SEQUENCE GOSUB 3Ø8Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLEUCL.

TEST PROGRAM NAME: SLEUCL	
Note:	
MERGE: SLEUCL.	
INPUT	OUTPUT
ENTER A,B 25,27 ENTER A,B 84,23 ENTER A,B	G.C.D OF 25 AND 27 = 1 L.C.M OF 25 AND 27 = 675 G.C.D OF 84 AND 23 = 1 L.C.M OF 84 AND 23 = 1932

**RATIONAL FRACTION CONVERSION
TO CONTINUED FRACTION**

SLRFC0.

To convert a rational fraction into a "regular" continued fraction i.e. one whose successive numerators are unity.

INPUT		
VARIABLE	DESCRIPTION	VALUE
PRFC01 PRFC02	numerator and denominator of rational fraction (positive integers)	
OUTPUT		
N PRFC01(N) FUN.	number of terms of continued fraction First term and successive denominators of continued fraction Return Status	$\emptyset$ = correct calculation 1 = PRFC01 and/or PRFC02 not positive integer
CALLING SEQUENCE GOSUB 31001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLRFCO.

TEST PROGRAM NAME: SLRFCO	
Note:	
MERGE: SLRFCO.	
INPUT	OUTPUT
ENTER A,B 151,119	REGULAR CONTINUED FRACTION OF 151/119 b0=1 b1=3 b2=1 b3=2 b4=1 b5=1 b6=4
ENTER A,B 771928,999999	REGULAR CONTINUED FRACTION OF 771928/999999 b0=0 b1=1 b2=3 b3=2 b4=1 b5=1 b6=1 b7=3 b11=1 b12=1 b13=1 b14=1 b15=1
ENTER A,B	

**QUADRATIC SURD CONVERSION
CONTINUED FRACTION**
SLSUC0.

To convert a quadratic surd into a "regular" continued fraction i.e.
one whose successive numerators are unity

INPUT		
VARIABLE	DESCRIPTION	VALUE
PSUC01	positive integer, $\sqrt{\text{PSUC01}}$ or quadratic surd	
PSUC02	number of terms of continued fraction to be calculated	
OUTPUT		
B(PSUC02)	successive terms of continued fraction	
FUN.	Return Status	$\emptyset$ = correct calculation 1 = PSUC01 and/or PSUC02 other than positive integers 2 = $\sqrt{\text{PSUC01}}$ not a quadratic surd $\sqrt{\text{PSUC01}}$ is returned in B(1)
CALLING SEQUENCE GOSUB 312 $\emptyset$ 1		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLSUCO.

TEST PROGRAM NAME: SLSUCO	
Note:	
MERGE: SLSUCO.	
INPUT	OUTPUT
ENTER SURD $^2 3$ ENTER NUMBER OF TERMS 5 ENTER SURD $^2 2$ ENTER NUMBER OF TERMS 5 ENTER SURD $^2 3$ ENTER NUMBER OF TERMS .5 ENTER SURD 2	REGULAR CONTINUED FRACTION OF 3^5 $b_0=1$ $b_1=1$ $b_2=2$ $b_3=1$ $b_4=2$ REGULAR CONTINUED FRACTION OF 2^5 $b_0=1$ $b_1=2$ $b_2=2$ $b_3=2$ $b_4=2$ POSITIVE INTEGERS ONLY

CONVERGENTS OF A CONTINUED FRACTION

To calculate the successive convergents of a continued fraction

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCFC01	To calculate the first convergent, PCFC01 is the first term of the continued fraction, b_0 . Otherwise PCFC01 = -1	
PCFC02	numerator and denominator of successive terms of the continued fraction	
PCFC03		
OUTPUT		
P2,Q2	numerator and denominator of the successive convergents	
FUN.	Return Status	Ø = correct calculation

CALCULATION P_0 , P_1 , Q_0 , Q_1
CALLING SEQUENCE GOSUB 31401
METHOD See Appendix A
CALLED SUBROUTINES

COMBINATORIAL ANALYSIS

TEST PROGRAM FOR SLCFCO.

TEST PROGRAM NAME: SLCFCO	
Note:	
MERGE: SLCFCO.	
INPUT	OUTPUT
ENTER b0 1 ENTER a1 b1 .45,1 ENTER a2 b2 .45,-2 ENTER a3 b3 .45,3 ENTER b0	b0=1 a1=.45 b1=1 CONVERGENT1=1.45/1=1.45 a2=.45 b2=-2 CONVERGENT2=2.45/-1.55=1.58064 516129032 a3=.45 b3=3 CONVERGENT3=-6.6975/-4.2=1.59464 285714286

FACTORIAL AND LOG FACTORIAL

To evaluate $N!$ and, using Stirling's Formula, to approximate the natural logarithm of $N!$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PFACT1 PFACT2	non-negative or positive integer depending on PFACT2 switch variable	1 = to calculate PFACT1! 2 = to calculate Ln PFACT1!
OUTPUT		
N1 FUN.	value of PFACT! or Ln PFACT1! depending upon the value of PFACT2 Return Status	Ø = correct calculation 1 = for PFACT2 = 1, PFACT1 other than a non-negative integer less than 171 for PFACT2 = 2, PFACT1 other than a positive integer 2 = PFACT2 other than 1 or 2

COMBINATORIAL ANALYSIS

CALLING SEQUENCE	GOSUB 31601
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLFACT.

TEST PROGRAM NAME: SLFACT	
Note:	
MERGE: SLFACT.	
INPUT	OUTPUT
ENTER N? 5 ENTER1(N!) OR 2(1nN!)? 1 ENTER N? 10 ENTER1(N!) OR 2(1nN!)? 2 ENTER N?	$5! = 1.200000000000D+002$ $1n 10! = 1.510441292822D+001$

BINOMIAL COEFFICIENTS

To generate the binomial coefficients $\binom{n}{r}$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PBIN01 PBIN02	parameters of $\binom{n}{r}$ For $0 \leq \text{PBIN02} \leq \text{PBIN01}$ a single coefficient is calculated For $\text{PBIN02} = -1$, $\binom{n}{r}$ is calculated for $r=0, 1 \dots \text{INT}(\frac{n}{2})$, where $\text{INT}(\frac{n}{2})$ is the integer part of $n/2$. The coefficients from $\text{INT}(n/2)+1$ to n are given by the relation $\binom{n}{r} = \binom{n}{n-r}$	
OUTPUT		
B B() FUN.	for $0 \leq \text{PBIN02} \leq \text{PBIN01}$, $B = \binom{n}{r}$ for $\text{PBIN02} = -1$, B() contains the $\text{INT}(\frac{n}{2})+1$ coefficients $\binom{n}{r}$, for $r=0, 1 \dots \text{INT}(\frac{n}{2})$, in ascending order of r Return Status	0 = correct calculation 1 = PBIN01 and/or PBIN02 not integers ≥ 0 (PBIN02=-1 excluded) 2 = PBIN02 > PBIN01
CALLING SEQUENCE GOSUB 31801		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLBINO.

[illegible]

MULTINOMIAL COEFFICIENTS

To generate the general multinomial coefficient $(n; n_1, n_2 \dots n_s)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PMULT1 N(PMULT1)	number of factors $n_1, n_2 \dots n_s$ vector of factors	
OUTPUT		
M N2 FUN.	Multinomial coefficient Sum of factors Return Status	0 = correct calculation 1 = number of factors not a positive integer
CALLING SEQUENCE GOSUB 32001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLMULT.

TEST PROGRAM NAME: SLMULT	
Note:	
MERGE: SLMULT.	
INPUT	OUTPUT
enter number of factors 6 enter factor 1 ? 18 enter factor 2 ? 11 enter factor 3 ? 6 enter factor 4 ? 3 enter factor 5 ? 2 enter factor 6 ? 2 enter number of factors?	number of factors 6 n1 = 18 n2 = 11 n3 = 6 n4 = 3 n5 = 2 n6 = 2 n = 42 multinomial coeff. = 3.18154257 170193D+23

COMBINATORIAL ANALYSIS

PROBABILITY OF DUPLICATION IN A GIVEN UNIVERSE

SLDUPL.

To evaluate the probability p_k , that at least two elements of a set K will be identical with respect to the same criteria, when there are n possible choices for each element

INPUT		
VARIABLE	DESCRIPTION	VALUE
PDUPL1	number of possible choices for each element (N)	
PDUPL2	number of elements in the set	
OUTPUT		
P(PDUPL1) P FUN.	probability of duplication for K = 1, 2...PDUPL1 probability of duplication for a given PDUPL2 Return Status	PDUPL2 = -1 PDUPL2 = 1,2,...PDUPL1 ∅ = correct calculation 1 = PDUPL1,PDUPL2 not positive integers (PDUPL1 = -1 excluded) with PDUPL1 ≥ PDUPL2
CALLING SEQUENCE GOSUB 32201		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLDUPL.

TEST PROGRAM NAME: SLDUPL	
Note:	
MERGE: SLDUPL.	
INPUT	OUTPUT
ENTER n,k ? 365, 20	n = 365 K = 20
ENTER n,k ? 365, 100	P _k = .41143838358058
ENTER n,k ? 10, -1	n ^k = 365 k = 100
	P = .999999692751072
	n ^k = 10 k = -1
	k P
	1 0 ^k
	2 .1
	3 .28
	4 .496
	5 .6976
	6 .8488
	7 .93952
	8 .981856
	9 .9963712
	10 .99963712
ENTER n,k ?	

4. ELEMENTARY FUNCTIONS

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of Elementary Functions.

CONTENTS

SLATN2.	Arctangent of a Ratio	4-1
SLCONV.	Reduction of an Angle to the First Quadrant	4-3
SLRPCC.	Rectangular to Polar Coordinates Conversion	4-5
SLPRCC.	Polar to Rectangular Coordinates Conversion	4-7

ARCTANGENT OF A RATIO

To compute the arctangent of a ratio, y/x of two real numbers in the range $(-\pi, \pi)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PATN 21 PATN 22	Denominator Numerator	
OUTPUT		
FUN.	The arctangent of PATN22/PATN21	
CALLING SEQUENCE GOSUB 32401		
METHOD		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLATN2.

TEST PROGRAM NAME: SLATN2	
Note:	
MERGE: SLATN2.	
INPUT	OUTPUT
ENTER x,y 0, 0 ENTER x,y 0, 1 ENTER x,y -1, 0 ENTER x,y	arctangent (0/0) = 0 arctangent (1/0) = 1.570796326795 arctangent (0/-1) = 3.14159265359

REDUCTION OF AN ANGLE TO THE FIRST QUADRANT

SLCONV.

To reduce an angle (measured in radians) to the range $(-\pi, \pi)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCONV1	angle in radians	
OUTPUT		
FUN.	The resultant value of the angle	
CALLING SEQUENCE GOSUB 32601		
METHOD		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLCONV.

TEST PROGRAM NAME: SLCONV	
Note:	
MERGE: SLCONV.	
INPUT	OUTPUT
ENTER ANGLE (radians) 1	angle transformed angle 1 1
ENTER ANGLE (radians) 2	2 2
ENTER ANGLE (radians) 3	3 3
ENTER ANGLE (radians) 4	4 -2.28318530718
ENTER ANGLE (radians) -3.1415927	-3.1415927 3.14159260718
ENTER ANGLE (radians)?	

RECTANGULAR TO POLAR COORDINATES
CONVERSION

SLRPCC.

To convert a complex number from rectangular to polar coordinates

INPUT		
VARIABLE	DESCRIPTION	VALUE
PRPCC1 PRPCC2	real part of the complex number imaginary part of the complex number	
OUTPUT		
P Q FUN.	modulus Phase Return Status	$-\pi < Q < \pi$ $\emptyset$ = correct calculation
CALLING SEQUENCE GOSUB 32801		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLRPCC.

TEST PROGRAM NAME: SLRPCC	
Note:	
MERGE: SLRPCC.	
INPUT	OUTPUT
ENTER x+iy As x,y -1, -1	x = -1 y = -1 modulus = 1.41421353816986 phase = 2.35619446833686
ENTER x+iy As x,y 1, 0	x = 1 y = 0 modulus = 1 phase = 0
ENTER x+iy As x,y 0, -100	x = 0 y = -100 modulus = 100 phase = -1.570796326795
ENTER x+iy As x,y	

POLAR TO RECTANGULAR COORDINATES
CONVERSION

SLPRCC.

To convert a complex number from polar to rectangular coordinates

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPRCC1 PPRCC2	modulus of the complex number phase of the complex number	
OUTPUT		
P1 P2 FUN.	real part of the complex number imaginary part of the complex number Return Status	Ø = correct calculation 1 = negative modulus
CALLING SEQUENCE GOSUB 33ØØ1		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLPRCC.

TEST PROGRAM NAME: SLPRCC	
Note:	
MERGE: SLPRCC.	
INPUT	OUTPUT
ENTER Modulus, Phase -1, -1	modulus = -1 phase = -1
ENTER Modulus, Phase 1000, 236	modulus >= 0 only modulus = 1000 phase = 236 x = -928.463339805603 y = -371.424198150635
ENTER Modulus, Phase 5, 0	modulus = 5 phase = 0 x = 5 y = 0
ENTER Modulus, Phase 0, 25	modulus = 0 phase = 25 x = 0 y = 0
ENTER Modulus, Phase	

5. ELEMENTARY FUNCTIONS (COMPLEX)

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of Complex Elementary Functions.

CONTENTS

SLCSIN.	Sine of a Complex Number	5-1	SLCZDZ.	Division of Two Complex Numbers	5-22
SLCCOS.	Cosine of a Complex Number	5-3	SLCSQR.	Square Root of a Complex Number	5-25
SLCTAN.	Tangent of a Complex Number	5-5	SLCZN.	Integral Power of a Complex Number (Z^n Recurrence)	5-27
SLCASN.	Arcsine of a Complex Number	5-7	SLCZA.	Real Power of a Complex Number	5-29
SLCACs.	Arcosine of a Complex Number	5-9			
SLCATN.	Arctangent of a Complex Number	5-11			
SLCLN.	Natural Logarithm of a Complex Number	5-13			
SLCEXP.	Exponential of a Complex Number	5-15			
SLCRZ.	Reciprocal of a Complex Number	5-17			
SLCZMZ.	Multiplication of Two Complex Numbers	5-19			

SINE OF A COMPLEX NUMBER

To compute the sine of a complex number, expressed either in rectangular or polar coordinates. The result is a complex number expressed in the same coordinates as the argument

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCSIN1	type of coordinates	Ø = rectangular 1 = polar
PCSIN2	real part of number if PCSIN1 = Ø modulus if PCSIN1 = 1	
PCSIN3	imaginary part of number if PCSIN1 = Ø phase if PCSIN1 = 1	
OUTPUT		
P	real part of sine if PCSIN1 = Ø modulus if PCSIN1 = 1	Ø = correct calculation 1 = negative modulus 2 = value of PCSIN1 is other than Ø or 1
Q	imaginary part of sine if PCSIN1 = Ø	
FUN.	phase if PCSIN1 = 1 Return Status	
CALLING SEQUENCE GOSUB 332Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 324Ø1		

TEST PROGRAM FOR SLCSIN.

TEST PROGRAM NAME: SLCSIN																						
Note:																						
MERGE: SLCSIN.																						
INPUT	OUTPUT																					
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y .52359878, 0 ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1, 0 ENTER 0 (rectangular) or 1 (polar)	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;"></td> <td style="width: 35%; text-align: center;">real part</td> <td style="width: 35%; text-align: center;">imaginary part</td> </tr> <tr> <td>z.</td> <td style="text-align: center;">5235987800000</td> <td style="text-align: center;">0.0000000</td> </tr> <tr> <td>sinz.</td> <td style="text-align: center;">5000000</td> <td style="text-align: center;">0.000000</td> </tr> <tr> <td></td> <td style="text-align: center;">modulus</td> <td style="text-align: center;">phase</td> </tr> <tr> <td>z</td> <td style="text-align: center;">1.0000000</td> <td style="text-align: center;">0.0000000</td> </tr> <tr> <td>sinz</td> <td style="text-align: center;">.8414709</td> <td style="text-align: center;">0.0000000</td> </tr> <tr> <td></td> <td style="text-align: center;">568024</td> <td></td> </tr> </table>		real part	imaginary part	z.	5235987800000	0.0000000	sinz.	5000000	0.000000		modulus	phase	z	1.0000000	0.0000000	sinz	.8414709	0.0000000		568024	
	real part	imaginary part																				
z.	5235987800000	0.0000000																				
sinz.	5000000	0.000000																				
	modulus	phase																				
z	1.0000000	0.0000000																				
sinz	.8414709	0.0000000																				
	568024																					

ELEMENTARY FUNCTIONS (COMPLEX)

SLCCOS.

COSINE OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCCOS1	type of coordinate	Ø = rectangular 1 = polar
PCCOS2	real part of number if PCCOS1 = Ø modulus if PCCOS = 1	
PCCOS3	imaginary part of number if PCCOS1 = Ø phase if PCCOS1 = 1	
OUTPUT		
P	real part of cosine if PCCOS1 = Ø modulus if PCCOS1 = 1	Ø = correct calculation 1 = modulus less than Ø 2 = PCCOS1 other than Ø or 1
Q	imaginary part of cosine if PCCOS1 = Ø phase if PCCOS1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 334Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 324Ø1		

TEST PROGRAM FOR SLCCOS.

TEST PROGRAM NAME: SLCCOS		
Note:		
MERGE: SLCCOS.		
INPUT	OUTPUT	
ENTER 0 (rectangular) or 1 (polar) 0		
ENTRY x+iy As x,y 00	z	real part imaginary part 0.000000 0.000000
	cosz	1.000000 0.000000
ENTER 0 (rectangular) or 1 (polar) 1		
ENTER modulus, phase 1, 1	z	modulus phase 1.000000000 1.000000000
	cosz	1.275615572 -.39067 9294 40248203
ENTER 0 (rectangular) or 1 (polar)		

TANGENT OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCTAN1	type of coordinates	Ø = rectangular 1 = polar
PCTAN2	real part of number if PCTAN1 = Ø modulus if PCTAN1 = 1	
PCTAN3	imaginary part of number if PCTAN1 = Ø phase if PCTAN1 = 1	
OUTPUT		
P	real part of tangent if PCTAN1 = Ø modulus if PCTAN1 = 1	Ø = correct calculation 1 = modulus less than Ø 2 = PCTAN1 other than Ø or 1
Q	imaginary part of tangent if PCTAN1 = Ø phase if PCTAN1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 336Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 324Ø1		

TEST PROGRAM FOR SLCTAN.

TEST PROGRAM NAME: SLCTAN																												
Note:																												
MERGE: SLCTAN.																												
INPUT	OUTPUT																											
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y 1, 0 ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1, 0	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 40%; text-align: right;">real part</td> <td style="width: 50%; text-align: right;">imaginary part</td> </tr> <tr> <td style="padding-right: 10px;">z</td> <td style="text-align: right;">1.000000</td> <td style="text-align: right;">0.000000</td> </tr> <tr> <td style="padding-right: 10px;">tanz</td> <td style="text-align: right;">1.557407</td> <td style="text-align: right;">0.000000</td> </tr> <tr> <td></td> <td style="text-align: right;">4957895</td> <td></td> </tr> <tr><td colspan="3"> </td></tr> <tr> <td></td> <td style="text-align: right;">modulus</td> <td style="text-align: right;">phase</td> </tr> <tr> <td style="padding-right: 10px;">z</td> <td style="text-align: right;">1.000000</td> <td style="text-align: right;">0.000000</td> </tr> <tr> <td style="padding-right: 10px;">tanz</td> <td style="text-align: right;">1.557407</td> <td style="text-align: right;">0.000000</td> </tr> <tr> <td></td> <td style="text-align: right;">4983597</td> <td></td> </tr> </table>		real part	imaginary part	z	1.000000	0.000000	tanz	1.557407	0.000000		4957895						modulus	phase	z	1.000000	0.000000	tanz	1.557407	0.000000		4983597	
	real part	imaginary part																										
z	1.000000	0.000000																										
tanz	1.557407	0.000000																										
	4957895																											
	modulus	phase																										
z	1.000000	0.000000																										
tanz	1.557407	0.000000																										
	4983597																											

ARCSINE OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCASN1	type of coordinates	Ø = rectangular 1 = polar
PCASN2	real part of number if PCASN1 = Ø modulus if PCASN1 = 1	
PCASN3	imaginary part of number if PCASN1 = Ø phase if PCASN1 = 1	
OUTPUT		
P	real part of arcsine if PCASN1 = Ø modulus if PCASN1 = 1	Ø = correct calculation 1 = modulus less than Ø 2 = PCASN1 other than Ø or 1
Q	imaginary part of arcsine if PCASN1 = Ø phase if PCASN1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 338Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 324Ø1		

TEST PROGRAM FOR SLCASN.

TEST PROGRAM NAME: SLCASN		
Note:		
MERGE: SLCASN.		
INPUT	OUTPUT	
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y .5, 0	real part	imaginary part
	z .50000000	0.000000
	arcsinz .5235987	0.000000
	901688	
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y 0, 1.175201193 6430	real part	imaginary part
	z 0.000000	1.17520119
	arcsinz 0.000000	36430
		.9999999
		403954
ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase .5, 0	modulus	phase
	z .50000000	0.000000
	arcsinz .52359879	0.000000
	01688	

ARCCOSINE OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCACS1	type of coordinates	Ø = rectangular 1 = polar
PCACS2	real part of number if PCACS1 = Ø modulus if PCACS1 = 1	
PCACS3	imaginary part of number if PCACS1 = Ø phase if PCACS1 = 1	
OUTPUT		
P	real part of arcsine if PCACS1 = Ø modulus if PCACS1 = 1	Ø = correct calculation 1 = modulus less than Ø 2 = PCACS1 other than Ø or 1
Q	imaginary part of arcsine if PCACS1 = Ø phase if PCACS1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 34001		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 32401		

TEST PROGRAM FOR SLCACS.

TEST PROGRAM NAME: SLCACS		
Note:		
MERGE: SLCACS.		
INPUT		OUTPUT
ENTER 0 (rectangular) or 1 (polar) 0		
ENTER x+iy As x,y 1, 0		
ENTER 0 (rectangular) or 1 (polar) 1		
ENTER modulus, phase .7, 0		
		real part imaginary part
		z 1.0000000 0.0000000
		arccosz 0.0000000 0.0000000
		modulus phase
		z .7000000 0.0000000
		arccosz .79539889 -.00030
		09721 69079539

ARCTANGENT OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCATN1	type of coordinate	Ø = rectangular 1 = polar
PCATN2	real part of number if PCATN1 = Ø modulus if PCATN1 = 1	
PCATN3	imaginary part of number if PCATN1 = Ø phase if PCATN1 = 1	
OUTPUT		
P	real part of arctangent if PCATN1 = Ø modulus if PCATN1 = 1	Ø = correct calculation 1 = modulus less than Ø 2 = PCATN1 other than 1 or Ø 3 = z ² = -1
Q	imaginary part of arctangent if PCATN1 = Ø phase if PCATN1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 342Ø1		
METHOD See Appendix		
CALLED SUBROUTINES GOSUB 324Ø1		

TEST PROGRAM FOR SLCATN.

TEST PROGRAM NAME: SLCATN	
Note:	
MERGE: SLCATN.	
INPUT	OUTPUT
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y 1.5574077246570, 0	real part imaginary part z 1.557407 0.0000000 7246570 arctanz 1.000000 0.0000000 00158934
ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1, 0	modulus phase z 1.0000000 0.000000 arctanz .7853981 0.000000 256485
ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1, 1.5707963267950	modulus phase z 1.0000000 1.57079632 67950 z*2 other than -1 only

NATURAL LOGARITHM OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCLN1	type of coordinates	Ø = rectangular 1 = polar
PCLN2	real part of number if PCLN1 = Ø modulus if PCLN1 = 1	
PCLN3	imaginary part of number if PCLN1 = Ø phase if PCLN1 = 1	
OUTPUT		
P	real part of logarithm if PCLN1 = Ø modulus if PCLN1 = 1	Ø = correct calculation 1 = modulus less than or equal to zero 2 = PCLN1 other than Ø or 1
Q	imaginary part of logarithm if PCLN1 = Ø phase if PCLN1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 344Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 324Ø1		

TEST PROGRAM FOR SLCLN.

TEST PROGRAM NAME: SLCLN		
Note:		
MERGE: SLCLN.		
INPUT	OUTPUT	
ENTER 0 (rectangular or 1 (polar) 0 ENTER x+iy As x,y 1, 0	z	real part 1.0000000 imaginary part 0.0000000
	1nz	0.0000000 0.0000000
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y 0, 0	z	real part 0.0000000 imaginary part 0.0000000
	modulus>0 only	
ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 0, 0	z	modulus 0.0000000 phase 0.0000000
	modulus>0 only	
ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase? 1, 0	z	modulus 1.0000000 phase 0.0000000
	1nz	0.0000000 0.0000000

ELEMENTARY FUNCTIONS (COMPLEX)

SLCEXP.

EXPONENTIAL OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCEXP1	types of coordinates	Ø = rectangular 1 = polar
PCEXP2	real part of number if PCEXP1 = Ø modulus if PCEXP1 = 1	
PCEXP3	imaginary part of number PCEXP1 = Ø phase if PCEXP1 = 1	
OUTPUT		
P	real part of exponential if PCEXP1 = Ø modulus if PCEXP1 = 1	Ø = correct calculation 1 = modulus less than Ø 2 = PCEXP1 other than Ø or 1
Q	imaginary part of exponential if PCEXP1 = Ø phase if PCEXP1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 346Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 326Ø1		

TEST PROGRAM FOR SLCEXP.

TEST PROGRAM NAME: SLCEXP																												
Note:																												
MERGE: SLCEXP.																												
INPUT	OUTPUT																											
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y 1, 0 ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1, 0	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 40%; text-align: center;">real part</td> <td style="width: 50%; text-align: center;">imaginary part</td> </tr> <tr> <td style="text-align: right;">z</td> <td style="text-align: center;">1.00000000</td> <td style="text-align: center;">0.00000000</td> </tr> <tr> <td style="text-align: right;">expz</td> <td style="text-align: center;">2.71828174</td> <td style="text-align: center;">0.00000000</td> </tr> <tr> <td></td> <td style="text-align: center;">59106</td> <td></td> </tr> <tr><td colspan="3"> </td></tr> <tr> <td></td> <td style="text-align: center;">modulus</td> <td style="text-align: center;">phase</td> </tr> <tr> <td style="text-align: right;">z</td> <td style="text-align: center;">1.00000000</td> <td style="text-align: center;">0.00000000</td> </tr> <tr> <td style="text-align: right;">expz</td> <td style="text-align: center;">2.7182174</td> <td style="text-align: center;">0.00000000</td> </tr> <tr> <td></td> <td style="text-align: center;">59106</td> <td></td> </tr> </table>		real part	imaginary part	z	1.00000000	0.00000000	expz	2.71828174	0.00000000		59106						modulus	phase	z	1.00000000	0.00000000	expz	2.7182174	0.00000000		59106	
	real part	imaginary part																										
z	1.00000000	0.00000000																										
expz	2.71828174	0.00000000																										
	59106																											
	modulus	phase																										
z	1.00000000	0.00000000																										
expz	2.7182174	0.00000000																										
	59106																											

RECIPROCAL OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCRZ1	type of coordinates	Ø = rectangular 1 = polar
PCRZ2	real part of number if PCRZ1 = Ø modulus if PCRZ1 = 1	
PCRZ3	imaginary part of number if PCRZ1 = Ø phase if PCRZ1 = 1	
OUTPUT		
P Q FUN.	real part of reciprocal if PCRZ1 = Ø modulus if PCRZ1 = 1 imaginary part of reciprocal if PCRZ1 = Ø Return Status	Ø = correct calculation 1 = modulus less than Ø 2 = PCRZ1 other than Ø or 1
CALLING SEQUENCE GOSUB 348Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 326Ø1		

TEST PROGRAM FOR SLCRZ.

TEST PROGRAM NAME: SLCRZ																			
Note:																			
MERGE: SLCRZ.																			
INPUT	OUTPUT																		
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y 0, 5 ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1, 2	<table> <tr> <td></td> <td>real part</td> <td>imaginary part</td> </tr> <tr> <td>z</td> <td>0.00000000</td> <td>5.00000000</td> </tr> <tr> <td>1/z</td> <td>0.00000000</td> <td>-.20000000</td> </tr> <tr> <td></td> <td>modulus</td> <td>phase</td> </tr> <tr> <td>z</td> <td>1.00000000</td> <td>2.00000000</td> </tr> <tr> <td>1/z</td> <td>1.00000000</td> <td>-2.00000000</td> </tr> </table>		real part	imaginary part	z	0.00000000	5.00000000	1/z	0.00000000	-.20000000		modulus	phase	z	1.00000000	2.00000000	1/z	1.00000000	-2.00000000
	real part	imaginary part																	
z	0.00000000	5.00000000																	
1/z	0.00000000	-.20000000																	
	modulus	phase																	
z	1.00000000	2.00000000																	
1/z	1.00000000	-2.00000000																	

MULTIPLICATION OF TWO COMPLEX NUMBERS

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCMZ5	type of coordinates	$\emptyset$ = rectangular 1 = polar
PCMZ1	real part of first number if PCMZ5 = $\emptyset$ modulus of first number if PCMZ5 = 1	
PCMZ2	imaginary part of first number if PCMZ5 = $\emptyset$ phase of first number if PCMZ5 = 1	
PCMZ3	real part of second number if PCMZ5 = $\emptyset$ modulus of second number if PCMZ5 = 1	
PCMZ4	imaginary part of second number if PCMZ5 = $\emptyset$ phase of second number if PCMZ5 = 1	

OUTPUT		
P	real part of the product if PCMZ5 = 0	0 = correct calculation 1 = modulus less than 0 2 = PCMZ5 other than 0 or 1
Q	modulus of the product if PCMZ5 = 1	
	imaginary part of the product if PCMZ5 = 0	
	phase of the product if PCMZ5 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 35001		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 32601		

ELEMENTARY FUNCTIONS (COMPLEX)

TEST PROGRAM FOR SLCZMZ.

TEST PROGRAM NAME: SLCZMZ		
Note:		
MERGE: SLCZMZ.		
INPUT	OUTPUT	
ENTER 0 (rectangular) or 1 (polar) 0		
ENTER x+iy As x,y for z1 ? 2, 2	z1	real part 2 imaginary part 2
ENTER x+iy As x,y for z2 ? 1, -4	z2	1 -4
	z1xz2	1.0000000 -6.000000000 0000000D+001 00000D+000
ENTER 0 (rectangular) or 1 (polar) 1		
ENTER modulus, phase for z1 ? 3, 6	z1	modulus 3 phase 6
ENTER modulus, phase for z2 ? 2, 3	z2	2 3
	z1*z2	6.0000000 2.716814692 0000000D+000 820D+000

DIVISION OF TWO COMPLEX NUMBERS

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCZDZ5	type of coordinates	$\emptyset$ = rectangular 1 = polar
PCZDZ1	real part of Z_1 if PCZDZ5 = $\emptyset$ modulus of Z_1 if PCZDZ5 = 1	$Z_1 = \text{PCZDZ1} + i \text{PCZDZ2}$
PCZDZ2	imaginary part of Z_1 if PCZDZ5 = $\emptyset$ phase of Z_1 if PCZDZ5 = 1	
PCZDZ3	real part of Z_2 if PCZDZ5 = $\emptyset$ modulus of Z_2 if PCZDZ5 = 1	$Z_2 = \text{PCZDZ3} + i \text{PCZDZ4}$
PCZDZ4	imaginary part of Z_2 if PCZDZ5 = $\emptyset$ phase of Z_2 if PCZDZ5 = 1	

ELEMENTARY FUNCTIONS (COMPLEX)

OUTPUT		
P	real part of ratio if PCZDZ5 = 0	0 = correct calculation 1 = $ Z_1 < 0$ or $ Z_2 \leq 0$ 2 = PCZDZ5 other than 0 or 1
Q	modulus of ratio if PCZDZ5 = 1	
	imaginary part of ratio if PCZDZ5 = 0	
FUN.	phase of ratio if PCZDZ5 = 1	
	Return Status	
CALLING SEQUENCE GOSUB 35201		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 32601		

TEST PROGRAM FOR SLCZDZ.

TEST PROGRAM NAME: SLCZDZ		
Note:		
MERGE: SLCZDZ.		
INPUT	OUTPUT	
ENTER 0 (rectangular) or 1 (polar) 0		
ENTER x+iy As x,y for z1 ? 2, 2	z1	real part imaginary part
ENTER x+iy As x,y for z2 ? 1, -4	z2	2 2
	z1/z2	1 -4
		-3.5294 5.8823529
		1176D-001 4D-001
ENTER 0 (rectangular) or 1 (polar) 1		
ENTER modulus, phase for z1 ? 10, 3.14	z1	modulus phase
ENTER modulus, phase for z2 ? 2, 3.14	z2	10 3.14
	z1/z2	2 3.14
		5.000000 0.000000
		000D+000 00E+000

SQUARE ROOT OF A COMPLEX NUMBER

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCSQR1	type of coordinates	$\emptyset$ = rectangular 1 = polar
PCSQR2	real part of number if PCSQR1 = $\emptyset$	
PCSQR3	imaginary part of number if PCSQR1 = $\emptyset$	
	modulus if PCSQR1 = 1	
	phase if PCSQR1 = 1	
OUTPUT		
P	real part of $\sqrt{Z}$ if PCSQR1 = $\emptyset$	Z = PCSQR2+iPCSQR3
Q	imaginary part of $\sqrt{Z}$ if PCSQR1 = $\emptyset$	
FUN.	Return Status	$\emptyset$ = correct calculation 1 = modulus less than $\emptyset$ 2 = PCSQR1 other than $\emptyset$ or 1
CALLING SEQUENCE GOSUB 354 $\emptyset$ 1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 326 $\emptyset$ 1		

TEST PROGRAM FOR SLCSQR.

TEST PROGRAM NAME: SLCSQR																									
Note:																									
MERGE: SLCSQR.																									
INPUT	OUTPUT																								
ENTER 0 (rectangular) or 1 (polar) 0 ENTER x+iy As x,y -1, 0 ENTER 0 (rectangular) or 1 (polar) 1 ENTER modulus, phase 1,-1.5707964	<table> <tr> <td></td> <td>real part</td> <td>imaginary part</td> </tr> <tr> <td>z</td> <td>-1</td> <td>0</td> </tr> <tr> <td>z^{.5}</td> <td>0.000000</td> <td>1.000000</td> </tr> <tr> <td></td> <td>00000E+000</td> <td>00000D+000</td> </tr> <tr> <td></td> <td>modulus</td> <td>phase</td> </tr> <tr> <td>z</td> <td>1</td> <td>-1.5707964</td> </tr> <tr> <td>z^{.5}</td> <td>1.000000</td> <td>-7.853982</td> </tr> <tr> <td></td> <td>00000D+000</td> <td>00000D-001</td> </tr> </table>		real part	imaginary part	z	-1	0	z ^{.5}	0.000000	1.000000		00000E+000	00000D+000		modulus	phase	z	1	-1.5707964	z ^{.5}	1.000000	-7.853982		00000D+000	00000D-001
	real part	imaginary part																							
z	-1	0																							
z ^{.5}	0.000000	1.000000																							
	00000E+000	00000D+000																							
	modulus	phase																							
z	1	-1.5707964																							
z ^{.5}	1.000000	-7.853982																							
	00000D+000	00000D-001																							

ELEMENTARY FUNCTIONS (COMPLEX)

INTEGRAL POWER OF A COMPLEX NUMBER (Z^n RECURRENCE)

SLCZN.

To compute the power Z^n with Z complex and n integer ≥ 0 . Both the argument and the results can be expressed either in rectangular or polar coordinates

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCZN1	type of coordinates	∅ = rectangular 1 = polar
PCZN2	exponent	Z = PCZN3+iPCZN4
PCZN3	real part of Z if PCZN1 = ∅ modulus of Z if PCZN1 = 1	
PCZN4	imaginary part of Z if PCZN1 = ∅ phase of Z if PCZN1 = 1	
OUTPUT		
P	real part of Z ⁿ if PCZN1 = ∅ modulus of Z ⁿ if PCZN1 = 1	∅ = correct calculation 1 = modulus less than ∅ 2 = exponent other than an integer ≥ ∅ 3 = PCZN1 other than ∅ or 1
Q	imaginary part of Z ⁿ if PCZN1 = ∅ phase of Z ⁿ if PCZN1 = 1	
FUN.	Return Status	
CALLING SEQUENCE GOSUB 356∅1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 326∅1		

TEST PROGRAM FOR SLCZN.

TEST PROGRAM NAME: SLCZN		
Note: In this test program if the parameter PCZMZ = -1 the base and/or the type of coordinate can be changed		
MERGE: SLCZN.		
INPUT	OUTPUT	
ENTER 0 (rectangular) or 1 (polar) 0		
ENTER x+iy As x,y 1, 1	z	real part imaginary part
ENTER EXPONENT 0	z^0	1 0
ENTER EXPONENT 1	z^1	1 1
ENTER EXPONENT 2	z^2	0 2
ENTER EXPONENT 3	z^3	-2 2
ENTER EXPONENT -1		
ENTER 0 (rectangular) or 1 (polar) 1		
ENTER modulus, phase 1, 1	z	modulus phase
ENTER EXPONENT 0	z^0	1 0
ENTER EXPONENT 1	z^1	1 1
ENTER 0 (rectangular) or 1 (polar)		

SLCZA.

REAL POWER OF A COMPLEX NUMBR

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCZA1	type of coordinates	∅ = rectangular 1 = polar
PCZA2	exponent	Z = PCZA3+iPCZA4
PCZA3	real part of Z if PCZA1 = ∅ modulus of Z if PCZA1 = 1	
PCZA4	imaginary part of Z if PCZA1 = ∅ phase of Z if PCZA1 = 1	
OUTPUT		
P	real part of Z ^a if PCZA1 = ∅ modulus of Z ^a if PCZA1 = 1	a(= PCZA2)
Q	imaginary part of Z ^a if PCZA1 = ∅ phase of Z ^a if PCZA1 = 1	∅ = correct calculation 1 = modulus less than ∅ 2 = PCZA1 other than ∅ or 1
FUN.	Return Status	
CALLING SEQUENCE GOSUB 358∅1		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 324∅1, 326∅1		

TEST PROGRAM FOR SLCZA.

TEST PROGRAM NAME: SLCZA		
Note: In this test program if the parameter PCZA2 = 9D99 the base and/or type of coordinate can be changed		
MERGE: SLCZA.		
INPUT	OUTPUT	
ENTER 0 (rectangular) or 1 (polar) 0	real part	imaginary part
ENTER x+iy As x,y 2, 0	z 2	0
ENTER EXPONENT .5	z^5 1.4142135	0
	3816986	
ENTER EXPONENT -1	z^-1 .5	0
ENTER EXPONENT 9D99		
ENTER 0 (rectangular) or 1 (polar) 1	modulus	phase
ENTER modulus, phase 1, 1	z 1	1
ENTER EXPONENT .25	z^.25 1	.25
ENTER 0 (rectangular) or 1 (polar)		

6. POLYNOMIALS

ABOUT THIS CHAPTER

This chapter contains subroutines for the Solution of Polynomials

CONTENTS

SLPLRC.	Evaluation of Real Polynomials (Complex Argument)	6-1
SLPLRR.	Evaluation of Real Polynomials (Real Argument)	6-3
SLPRRR.	Calculating the Coefficients of a Polynomial from Roots	6-5
SLPLYM.	Multiplication of Two Real Polynomials	6-7
SLPLYD.	Division of Two Real Polynomials	6-9
SLPTRA.	Translation of Coefficients of a Real Polynomial	6-12

POLYNOMIALS

EVALUATION OF REAL POLYNOMIALS (COMPLEX ARGUMENT)

SLPLRC.

To evaluate a polynomial with real coefficients and complex arguments

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPLRC1 PPLRC2 PPLRC3 E(PPLRC3+1)	real part of the argument imaginary part of the argument degree of polynomial vector of coefficients (in descending order)	
OUTPUT		
P Q FUN.	real part of the value imaginary part of the value Return Status	$\emptyset$ = correct calculation 1 = degree other than a positive integer
CALLING SEQUENCE GOSUB 36001		
METHOD		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLPLRC.

TEST PROGRAM NAME: SLPLRC	
Note: In this test program if the real part of the argument (PPLRC1) = 9D99 it is possible to change the degree of the polynomial and then the values of the coefficients	
MERGE: SLPLRC.	
INPUT	OUTPUT
ENTER DEGREE OF POLYNOMIAL ? 4 COEFFICIENTS (in descending order) ENTER C4 ? 1 ENTER C3 ? 2 ENTER C2 ? 3 ENTER C1 ? 4 ENTER C0 ? 5 ENTER ARGUMENT x+iy As x,y ? 1, 14 ENTER ARGUMENT x+iy As x,y ? 0, 1 ENTER ARGUMENT x+iy As x,y ? 9D99, 0 ENTER DEGREE OF POLYNOMIAL?	DEGREE OF POLYNOMIAL IS 4 $p(1+14i) = 35491 - 16184i$ $p(0+1i) = 3+2i$

**EVALUATION OF REAL POLYNOMIALS
(REAL ARGUMENT)**

SLPLRR.

To calculate the value of a polynomial with real coefficients and real arguments

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPLRR1 PPLRR2 E(PPLRR2+1)	argument degree vector of coefficients (in descending order)	
OUTPUT		
P FUN.	value of polynomial Return Status	0 = correct calculation 1 = degree other than a positive integer
CALLING SEQUENCE GOSUB 36201		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLPLRR.

TEST PROGRAM NAME: SLPLRR	
Note: In this test program if the argument (PPLRR1) = 9D99 it is possible to change the degree of the polynomial and the values of the coefficients	
MERGE: SLPLRR.	
INPUT	OUTPUT
ENTER DEGREE ? 3 COEFFICIENTS (in descending order): ENTER COEFFICIENT C3 ? 1 ENTER COEFFICIENT C2 ? -6 ENTER COEFFICIENT C1 ? 11 ENTER COEFFICIENT C0 ? -6 ENTER ARGUMENT ? 1 ENTER ARGUMENT ? 2 ENTER ARGUMENT ? 3 ENTER ARGUMENT ? 4 ENTER ARGUMENT ? 3.1415927 ENTER ARGUMENT ? 9D99 ENTER DEGREE ?	DEGREE OF POLYNOMIAL IS 3 P(1) = 0 P(2) = 0 P(3) = 0 P(4) = 6 P(3.1415927) = .346169598291242

POLYNOMIALS

CALCULATING THE COEFFICIENTS OF A POLYNOMIAL FROM ROOTS

SLPRRR.

To compute the coefficients of a real polynomial from its roots

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPRRR1 E(PPRRR1+1)	number of roots vector of roots	
OUTPUT		
A(PPRRR1+1) FUN.	vector of coefficients (in descending order) Return Status	$\emptyset$ = correct calculation 1 = number of roots other than a positive integer
CALLING SEQUENCE GOSUB 364 $\emptyset$ 1		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLPRRR.

TEST PROGRAM NAME: SLPRRR	
Note:	
MERGE: SLPRRR.	
INPUT	OUTPUT
ENTER #ROOTS ? 3 ENTER ROOT 1 ? 1 ENTER ROOT 2 ? 2 ENTER ROOT 3 ? 3 ENTER #ROOTS ? 5 ENTER ROOT 1 ? -1 ENTER ROOT 2 ? 1 ENTER ROOT 3 ? 3 ENTER ROOT 4 ? 5 ENTER ROOT 5 ? 7 ENTER #ROOTS ?	ROOTS: 3 DEGREE OF POLYNOMIAL IS 3 COEFFICIENTS (IN DESCENDING ORDER) 1 -6 11 -6 ROOTS: 5 DEGREE OF POLYNOMIAL IS 5 COEFFICIENTS (in descending order) 1 -15 70 -90 -71 105

MULTIPLICATION OF TWO REAL POLYNOMIALS

To compute the product of two polynomials with real coefficients

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPLYM1 PPLYM2 E() 		

TEST PROGRAM FOR SLPLYM.

TEST PROGRAM NAME: SLPLYM	
Note:	
MERGE: SLPLYM.	
INPUT	OUTPUT
ENTER DEGREE OF POLYNOMIAL 1 ? 3 COEFFICIENTS (in descending order): ENTER a3 ? 1 ENTER a2 ? -6 ENTER a1 ? 11 ENTER a0 ? -6 ENTER DEGREE OF POLYNOMIAL 2 ? 5 COEFFICIENTS (in descending order): ENTER a5 ? 1 ENTER a4 ? 1 ENTER a3 ? -8 ENTER a2 ? -16 ENTER a1 ? 7 ENTER a0 ? 15 ENTER DEGREE OF POLYNOMIAL 1 ?	DEGREE OF POLYNOMIAL1 IS 3 DEGREE OF POLYNOMIAL2 IS 5 DEGREE OF PRODUCT IS 8 COEFFICIENTS (in descending order) 1 -5 -3 37 9 -155 83 123 -90

DIVISION OF TWO REAL POLYNOMIALS

To compute the quotient and remainder of the division of two polynomials with real coefficients

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPLYD1 PPLYD2 E() F()	degree of dividend degree of divisor, $PPLYD2 \leq PPLYD1$ array of coefficients of dividend in increasing order, $E(1) = a_0 \dots E(l) = a_{i-1}$ array of coefficients of divisor	
OUTPUT		
P() N Q() Q1 FUN.	array of coefficients of quotient degree of quotient array of coefficients of remainder degree of remainder, $Q1 = -1$ if there is no remainder Return Status	$\emptyset$ = correct calculation 1 = degree of dividend or divisor other than positive integers with $PPLYD2 \leq PPLYD1$ 2 = leading coefficient of divisor is zero

CALLING SEQUENCE	GOSUB 36801
METHOD	See Appendix A
CALLED SUBROUTINES	

POLYNOMIALS

TEST PROGRAM FOR SLPLYD.

[illegible]

TRANSLATION OF COEFFICIENTS OF A REAL POLYNOMIAL

SLPTRA.

Given the real polynomial $p(x)$, to calculate the coefficients of $q(x) = p(x+s)$ where s is a real constant. S is a root of $p(x)$ of multiplicity j if the coefficients of $x^{j-1}, x^{j-2} \dots x^0$ of $q(x)$ are all zero

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPTRA1 PPTRA2 E(PPTRA1+1)	degree of $p(x)$ shift coefficients of $p(x)$ in increasing order	
OUTPUT		
E(PPTRA1+1) FUN.	coefficients of $q(x) = p(x+s)$ in increasing order Note: $p(x)$ is destroyed and replaced by $p(x+s)$ Return Status	$\emptyset$ = correct calculation 1 = degree of $p(x)$ not a positive integer
CALLING SEQUENCE GOSUB 37001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLPTRA.

6-13

CC

C

C

C

CC

7. HIGHER MATHEMATICAL FUNCTIONS

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of problems in Higher Mathematical Functions.

CONTENTS

SLKMF.	Complete Elliptic Integral of First Kind	7-1	SLIIX.	Definite Integral of the Bessel Function $I_0(x)$	7-19
SLEMF.	Complete Elliptic Integral of Second Kind	7-3	SLSF.	Fresnel Integral $S(x)$	7-21
			SLCF.	Fresnel Integral $C(x)$	7-23
SLLAGG.	Generalised Laguerre Poly- nomial $L_n^{(a)}(x)$	7-5	SLCHYF.	Confluent Hyper- geometric Function	7-25
SLHNF.	Hermite Polynomial $H_n(x)$	7-7	SLGHYP.	Gauss Hyper- geometric Function	7-27
SLHEN.	Hermite Polynomial $He_n(x)$	7-9	SLSIF.	Sine Integral $Si(x)$	7-29
SLFOUR.	Evaluation of Fourier Series	7-11	SLCINF.	Cosine Integral $Cin(x)$	7-31
SLGAMA.	Gamma Function	7-13	SLEIF.	Exponential Integral $Ei(x)$	7-33
SLERF.	Error Function $erf(x)$	7-15	SLEINF.	Exponential Integral $Ei(x)$	7-35
SLBJN.	Bessel Function of Integer Order $J_n(x)$	7-17			

COMPLETE ELLIPTIC INTEGRAL OF FIRST KIND

SLKMF.

Evaluation of the Complete Elliptic Integral of the First Kind

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PKMF1	argument	
OUTPUT		
F Q FUN.	value of function value of last term calculated Return Status	 0 = correct calculation 1 = no convergence after 1000 terms 2 = $ x \geq 1$
CALLING SEQUENCE GOSUB 37201		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLKMF.

TEST PROGRAM NAME: SLKMF	
Note:	
MERGE: SLKMF.	
INPUT	OUTPUT
ENTER X ? -.9	-.9 1.329362192885
ENTER X ? -.55	-.55 1.403573750272
ENTER X ? 0	0 1.570796326795
ENTER X ? .3333	.3333 1.733896307695
ENTER X ? .999999	* ! last term !
	* 3.17911319855089D-04
	. 999999 5.128653257586
ENTER X ?	

**COMPLETE ELLIPTIC INTEGRAL
OF SECOND KIND**

SLEMF.

Evaluation of the Complete Elliptical Integral of the Second Kind

$$E(x) = \int_0^{\pi/2} \sqrt{1 - x \sin^2 \vartheta} \, d\vartheta$$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PEMF1	argument	
OUTPUT		
F Q FUN.	value of function value of last term evaluated Return Status	$\emptyset$ = correct calculation 1 = $ x \geq 1$ 2 = no convergence after 1000 terms
CALLING SEQUENCE GOSUB 37401		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLEMF.

TEST PROGRAM NAME: SLEMF	
Note:	
MERGE: SLEMF.	
INPUT	OUTPUT
ENTER X ? 0	X Complete elliptic
ENTER X ? .1	0 1.570796326795
ENTER X ? .3333	.1 1.530757636899
ENTER X ? -.9	.3333 1.430330437176
ENTER X ? -.999999	-.9 1.879834731649
	* !last term!
	* 3.15062939249677D-04
	-.99999 1.910095773870
ENTER X ?	

GENERALIZED LAGUERRE POLYNOMIAL
 $L_n^{(a)}(x)$

SLLAGG.

Evaluation of the Generalised Laguerre Polynomial $L_n^{(a)}(x)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PLAGG1 PLAGG2 PLAGG3	degree (n) parameter, a, of $L_n^{(a)}(x)$ argument	
OUTPUT		
L FUN.	value of $L_n^{(a)}(x)$ Return Status	$\emptyset$ = correct calculation 1 = degree other than a non-negative integer
CALLING SEQUENCE GOSUB 37601		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLLAGG.

TEST PROGRAM NAME: SLLAGG					
Note: In this test program if the argument (PLAGG3) = 9D99 it is possible to change the degree and the parameter a of the polynomial					
MERGE: SLLAGG.					
INPUT		OUTPUT			
		x	a	n	generalised Laguerre poly- nomial $L(a)_n(x)$
ENTER a,n? 1.5, 3					
ENTER X ? 0		0	1.5	3	6.562500000000 00D+000
ENTER X ? 1		1	1.5	3	7.708333333 333D-001
ENTER X ? 9D99					
ENTER a,n ?					

HERMITE POLYNOMIAL $H_n(x)$ Evaluation of the Hermite Polynomial $H_n(x)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PHNF1 PHNF2	degree (n) argument (x)	
OUTPUT		
H FUN.	value of $H_n(x)$ Return Status	$\emptyset$ = correct calculation 1 = degree not a integer $\geq \emptyset$
CALLING SEQUENCE GOSUB 37801		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLHNF.

TEST PROGRAM NAME: SLHNF		
Note: In this test program if the argument (PHNF2) = 9D99 it is possible to change the degree of the polynomial		
MERGE: SLHNF.		
INPUT	OUTPUT	
ENTER n ? 2 ENTER X ? 1 ENTER X ? 2.5 ENTER X ? -.1 ENTER X ? 9D99 ENTER n ?	x 1 2.5 -.1	n 2 2 2 2 2 2
	Hermite Polynomial Hn(x) 2.000000000000000D+000 2.300000000000000D+001 -1.960000000000000D+000	

HERMITE POLYNOMIAL $He_n(x)$ Evaluation of the Hermite Polynomial $He_n(x)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PHEN1 PHEN2	degree (n) argument (x)	
OUTPUT		
H FUN.	value of $He_n(x)$ Return Status	$\emptyset$ = correct calculation 1 = degree not an integer $\geq \emptyset$
CALLING SEQUENCE GOSUB 38001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLHEN.

TEST PROGRAM NAME: SLHEN		
Note: In this test program if the argument (PHEN2) = 9D99 it is possible to change the degree of the polynomial		
MERGE: SLHEN.		
INPUT	OUTPUT	
ENTER n ? 2 ENTER X ? 0 ENTER X ? -2 ENTER X ? 9D99 ENTER n ?	x	n Hermite Polynomial Hen(x)
	0	2 -1.000000000000D+0000
	-2	2 3.000000000000D+0000

EVALUATION OF FOURIER SERIES

Evaluation of a Fourier series

INPUT		
VARIABLE	DESCRIPTION	VALUE
PFOUR1	maximum harmonic order to be considered	(See Appendix A)
PFOUR2	value for which the series will be evaluated	
A(PFOUR1+1)	array of A values	
B (N)	array of B values	
OUTPUT		
F FUN.	value of the function Return Status	$\emptyset$ = correct calculation 1 = harmonic order not an integer $\geq \emptyset$
CALLING SEQUENCE GOSUB 382 $\emptyset$ 1		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLFOUR.

TEST PROGRAM NAME: SLFOUR	
Note: In this test program if the value of the argument (PFOUR2) = 9D99 the Harmonic Order and values of the coefficients can be changed	
MERGE: SLFOUR.	
INPUT	OUTPUT
ENTER HARMONIC ORDER ? 5 COEFFICIENTS: ENTER A5, B5 ? 1,1 ENTER A4, B4 ? 2,2 ENTER A3, B3 ? 3,3 ENTER A2, B2 ? 4,4 ENTER A1, B1 ? 5,5 ENTER A0 ? 6 ENTER X ? 1 ENTER X ? 2 ENTER X ? 9 ENTER X ? 9D99 ENTER HARMONIC ORDER ?	F(1) = 5.83869786560535 F(2) = 2.17085722088814 F(9) = 1.27171117067337

GAMMA FUNCTION

To evaluate the gamma function $\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt \quad (x > 0)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PGAMA1	value for which the function will be calculated	
OUTPUT		
FUN.	the value of $\Gamma(x)$. For $x = 0$, -1, -2, ...-n a value of 9.D99 is returned	
CALLING SEQUENCE GOSUB 38401		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLGAMA.

TEST PROGRAM NAME: SLGAMA											
Note:											
MERGE: SLGAMA.											
INPUT	OUTPUT										
ENTER X ? 1 ENTER X ? -1 ENTER X ? 2.5 ENTER X ? -2 ENTER X ?	<table> <tr> <td>x</td><td>Gamma Function</td></tr> <tr> <td>1</td><td>1.000000000000D+000</td></tr> <tr> <td>-1</td><td>-3.544907983672D+000</td></tr> <tr> <td>2.5</td><td>1.329340493877D+000</td></tr> <tr> <td>-2</td><td>9.000000000000D+099</td></tr> </table>	x	Gamma Function	1	1.000000000000D+000	-1	-3.544907983672D+000	2.5	1.329340493877D+000	-2	9.000000000000D+099
x	Gamma Function										
1	1.000000000000D+000										
-1	-3.544907983672D+000										
2.5	1.329340493877D+000										
-2	9.000000000000D+099										

SLERF.

ERROR FUNCTION erf(x)

To calculate the value of the error function $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PERF1	value for which the function is to be calculated	
OUTPUT		
FUN.	The value of erf(x)	
CALLING SEQUENCE GOSUB 38601		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLERF.

TEST PROGRAM NAME: SLERF	
Note:	
MERGE: SLERF.	
INPUT	OUTPUT
ENTER X ? 0	x error function
ENTER X ? 1	0.000000000000E+000
ENTER X ? 4.9	1 8.427007477117D-001
ENTER X ? -100	4.9 9.999998981192D-001
ENTER X ?	-100 -1.000000000000D+000

SLBJN.

BESSEL FUNCTION OF INTEGER ORDER $J_n(x)$

To evaluate the function y , where $J_n(x)$ is a solution of the differential equation

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PBJN1 PBJN2	order (n) argument (x)	
OUTPUT		
J FUN.	value of $J_n(x)$ Return Status	0 = correct calculation 1 = order not an integer
CALLING SEQUENCE GOSUB 38801		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLBJN.

TEST PROGRAM NAME: BJJ		
Note: In this test program if the argument (PBJN2) = 9D99 it is possible to change the value of the order (PBJN1)		
MERGE: SLBJN.		
INPUT	OUTPUT	
	X	n Bessel Function of integer order Jn(x)
ENTER n ? 0		
ENTER X ? .1	.1	0 9.975015620660D-001
ENTER X ? .5	.5	0 9.384698072402D-001
ENTER X ? 9D99		
ENTER n ? 3		
ENTER X ? 2	2	3 1.289432494744D-001
ENTER X ?		

SLIIOX.

DEFINITE INTEGRAL OF THE BESSEL FUNCTION $I_0(x)$

To evaluate the integral $f(x) = \int_0^x I_0(t) dt$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PIIOX1	Superior limit of integration	
OUTPUT		
FUN.	value of the function	
CALLING SEQUENCE GOSUB 39001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLIIØX.

TEST PROGRAM NAME: INT	
Note:	
MERGE: SLIIØX.	
INPUT	OUTPUT
ENTER X ? Ø	X definite integral of
ENTER X ? .1	Bessel Function Io(x)
ENTER X ? 5Ø	Ø Ø
ENTER X ?	.1 .1ØØØ83364589534
	5Ø 2.96296592994721D+2Ø

FRESNEL INTEGRAL S(x)

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PSF1	upper limit of integration	
OUTPUT		
S. FUN.	the value of the function Return Status	$\emptyset$ = correct calculation $1 = x \geq 3.5$
CALLING SEQUENCE GOSUB 39201		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLSF.

TEST PROGRAM NAME: SFR	
Note:	
MERGE: SLSF.	
INPUT	OUTPUT
ENTER X ? 0 ENTER X ? -.5 ENTER X ? 3.3 ENTER X ?	X Fresnel Integral S(x) 0 0 -.5 -6.47324328600034D-02 3.3 .519286079733048

FRESNEL INTEGRAL C(x)

To evaluate the integral $C(x) = \int_0^x \cos\left(\frac{\pi}{2} t^2\right) dt$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCF1	superior limit of integration	
OUTPUT		
C FUN.	the value of the function Return Status	$\emptyset$ = correct calculation $1 = x \geq 3.5$
CALLING SEQUENCE GOSUB 39401		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLCF.

TEST PROGRAM NAME: CFR	
Note:	
MERGE: SLCF.	
INPUT	OUTPUT
ENTER X ? 0 ENTER X ? -1 ENTER X ? 3.1 ENTER X ?	X Fresnel Integral C(x) 0 0 -1 -.779893387273324 3.1 .561593949173428

CONFLUENT HYPERGEOMETRIC FUNCTION

Evaluation of the confluent hypergeometric function

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCHYF1 PCHYF2 PCHYF3	parameters a,b and argument x of $M(a,b,x)$	See Appendix A
OUTPUT		
S FUN.	value of $M(a,b,x)$ Return Status	$\emptyset$ = correct calculation 1 if $b = \emptyset, -1, -2\dots$
CALLING SEQUENCE GOSUB 39601		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLCHYF.

TEST PROGRAM NAME: CHYF			
Note: In this program if the argument (PCHYF3) = 9D99 it is possible to change the values of the other two parameters			
MERGE: SLCHYF.			
INPUT	OUTPUT		
ENTER a,b ? -.5, .2 ENTER X ? .3 ENTER X ? .4 ENTER X ? 9D99 ENTER a,b ? -1, .6 ENTER X ? -5 ENTER X ?	x	a	b Confluent hypergeometric function M(a,b,x)
	.3	-.5	.2 1.997319258826D -001
	.4	-.5	.2 -9.15428012303 7D-002
	-5	-1	.6 9.3333333333D+ 000

GAUSS HYPERGEOMETRIC FUNCTION

Evaluation of the Gauss hypergeometric function

INPUT		
VARIABLE	DESCRIPTION	VALUE
PGHYP1 PGHYP2 PGHYP3 PGHYP4	parameters a,b,c and argument x of $F(a,b,c,x)$	See Appendix A
OUTPUT		
S FUN.	value of function Return Status	$\emptyset$ = correct calculation 1 = incorrect parameter c 2 = invalid x for a,b,c 3 = no convergence af- ter 1000 terms
CALLING SEQUENCE GOSUB 39801		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLGHYP.

TEST PROGRAM NAME: GHYP	
Note: In this program if the argument (PGHYP4) = 9D99 it is possible to change the values of the other parameters	
MERGE: SLGHYP.	
INPUT	OUTPUT
ENTER a,b,c ? 2,2,4.5 ENTER X ? 5 ENTER X ? 9D99 ENTER a,b,c ? -1.5, -2.5, 5 ENTER X ? 1 ENTER X ?	x a b c Gauss Hyper geometric function F(a,b, c,x) .5 2 2 4.5 1.755180915381D +000 1 -1.5 -2.5 5 1.796302591496D +000

SLSIF.

SINE INTEGRAL Si(x)

To evaluate $\text{Si}(x) = \int_0^x \frac{\sin t}{t} dt$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PSIF1	value for which the function will be calculated	
OUTPUT		
FUN.	value of function	
CALLING SEQUENCE GOSUB 40001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLSIF.

TEST PROGRAM NAME: SIF	
Note:	
MERGE: SLSIF.	
INPUT	OUTPUT
ENTER X ? .5 ENTER X ? 1.5 ENTER X ? 10 ENTER X ?	X sine integral Si(x) .5 4.931074180431D-001 1.5 1.324683531172D+000 10 1.658347594219D+000

SLCINF.

COSINE INTEGRAL Cin(x)

To calculate $\text{Cin}(x) = \int_0^x \frac{(1 - \cos t)}{t} dt$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCINF1	upper limit of integration	
OUTPUT		
FUN.	value of function	
CALLING SEQUENCE GOSUB 40201		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLCINF.

TEST PROGRAM NAME: CINF	
Note:	
MERGE: SLCINF.	
INPUT	OUTPUT
ENTER X ? -.5	X cosine integral Cin(x) -.5 6.185256314820D-000
ENTER X ? 10	10 2.925257190900D+000
ENTER X ?	

SLEIF.

EXPONENTIAL INTEGRAL $Ei(x)$

To evaluate $Ei(x) = \int_{-x}^{\infty} \frac{e^{-t}}{t} dt \quad (x > 0)$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PEIF1	inferior limit of integration (>0)	
OUTPUT		
E FUN.	value of function Return Status	0 = correct calculation $1:x \leq 0$
CALLING SEQUENCE GOSUB 40401		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLEIF.

TEST PROGRAM NAME: EIF	
Note:	
MERGE: SLEIF.	
INPUT	OUTPUT
ENTER X ? .1 ENTER X ? 1.7 ENTER X ? 5 ENTER X ?	X exponential integral Ei(x) .1 -6.22812845946D+000 1.7 3.920963214242D+000 5 4.018527538587D+001

SLEINF.

EXPONENTIAL INTEGRAL Ein(x)

To calculate the value of

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PEINF1	inferior limit of integration(x)	
OUTPUT		
FUN.	the value of the function	
CALLING SEQUENCE GOSUB 47001		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLEINF.

TEST PROGRAM NAME: EINF	
Note:	
MERGE: SLEINF.	
INPUT	OUTPUT
	X exponential integral Ein(x)
ENTER X ? .1	.1 .97554530327D-001
ENTER X ? 3	3 .16888763347D+001
ENTER X ? 5	5 .21878018729D+001
ENTER X ?	

8. SOLUTION OF EQUATIONS

ABOUT THIS CHAPTER

This chapter contains subroutines for the Solution of Equations.

CONTENTS

SLBAIR.	Roots of a Real Polynomial (Newton- Bairstow)	8-1
SLRBIS.	Roots of a Real Function (Bisection)	8-4
SLNLIN.	Solution of a Non- linear System	8-7

SOLUTION OF EQUATIONS

ROOTS OF A REAL POLYNOMIAL (NEWTON - BAIRSTOW)

SLBAIR.

The routine calculates real or complex roots of a polynomial with real coefficients

INPUT		
VARIABLE	DESCRIPTION	VALUE
PBAIR1	first approximation to p_k	$\emptyset$ = no deflation 1 = deflation
PBAIR2	first approximation to q_k	
PBAIR3	tolerance(E)	
PBAIR4	maximum number of iterations (Z)	
PBAIR5	deflation switch for PBAIR5 = 1 the original polynomial $f(x)$ will be deflated by a found factor $g(x)$. The coefficients of $f(x)/g(x)$ will replace those of $f(x)$ in the vector E(). Repeated application of the routine with PBAIR5 = 1 will find all roots of $f(x)$	
D	degree of polynomial $f(x)$	
E(D+1)	coefficients of $f(x)$ in ascending order	

OUTPUT		
R1,R2 or R1 or R1 R2 P1,P2 Q1,Q2 FUN.	two real roots of $f(x)$ one real root of $f(x)$ real part of complex imaginary part conjugates real and imaginary parts of value of polynomial at complex roots, or value at each real root last computed values of P_k, Q_k Return Status	$\emptyset$ = only one real root R1,R2=$\emptyset$ 1 = two real roots 2 = two complex roots 3 = no solution after PBAIR4 iterations 4 = degree other than positive integer
CALLING SEQUENCE GOSUB 40601		
METHOD See Appendix A		
CALLED SUBROUTINES GOSUB 36001		

SOLUTION OF EQUATIONS

TEST PROGRAM FOR SLBAIR.

TEST PROGRAM NAME: BAIR	
Note: In this program if the approximations to P_k (PBAIR1) = 9D99 it is possible to change the value of the tolerance (PBAIR3); and if the tolerance = 9D99 it is possible to change the values of the degree of the polynomial and the coefficients	
MERGE: SLBAIR.	
INPUT	OUTPUT
ENTER DEGREE ? 5 COEFFICIENTS (in descending order): ENTER C5 ? 1 ENTER C4 ? 1 ENTER C3 ? -8 ENTER C2 ? -16 ENTER C1 ? 7 ENTER C0 ? 15 ENTER TOLERANCE ? .000001 ENTER MAX # OF ITERATIONS ? 50 DEFLATION ? 1(YES),0(NO) ? 1 ENTER APPROXIMATION p,q ? -0, -0 ENTER APPROXIMATION p,q ? 9D99, 0 ENTER TOLERANCE ? 9D99 ENTER DEGREE ?	DEGREE OF POLYNOMIAL IS 5 C5 = 1 C4 = 1 C3 = -8 C2 = -16 C1 = 7 C0 = 15 TOLERANCE = .000001 MAX # OF ITERATIONS = 50 DEFLATION REQUESTED APPROXIMATION FACTOR p=0 q=0 TWO REAL ZEROS .9999999, -1.0000000 01 FACTOR p=-00000002 q=1.00000001 VALUES OF POLYNOMIAL 4.00000001121725D-07 -1.600000000803961D-07

ROOTS OF A REAL FUNCTION (BISECTION)

To locate a root of a real function by bisection of a user-defined interval

INPUT		
VARIABLE	DESCRIPTION	VALUE
PRBIS1 Q1,Q2	tolerance start and end of interval to be examined	
OUTPUT		
U Q1,Q2 FUN.	root of $f(x)$ final extremes of interval Return Status	$\emptyset$ = root found 1 = a pole is found in the interval Q1,Q2 2 = no root in interval 3 = tolerance not positive
CALLING SEQUENCE GOSUB 40801		
METHOD See Appendix A		
CALLED SUBROUTINES		

SOLUTION OF EQUATIONS

Note

The function $f(x)$ must be included in the calling program as FNB. For example, $f(x) = e^x - x$ would be coded as `DEF FNB(x) = EXP(x) - x`

TEST PROGRAM FOR SLRBIS.

TEST PROGRAM NAME: BISEL	
Note:	
MERGE: SLRBIS	
INPUT	OUTPUT
ENTER start,end of INTERVAL ? 0, 1 ENTER tolerance ? .001 ENTER start,end of INTERVAL ? 1, 10 ENTER tolerance ? .001 ENTER start,end of INTERVAL ?	interval [0,1] tolerance = .001 a solution is x = .56689453125 f(x) = 3.8927398147583D-04 interval [1,10] tolerance = .001 there is no root in the given interval

SOLUTION OF A NON-LINEAR SYSTEM

Computation of a solution vector $(\alpha_1, \dots, \alpha_n)$ of the system

$F_1(x_1, x_2, \dots, x_n), \dots, F_n(x_1, x_2, \dots, x_n)$ i.e. an n-tuple $(\alpha_1, \dots, \alpha_n)$ such that:

$$F_1(\alpha_1, \dots, \alpha_n) = F_2(\alpha_1, \dots, \alpha_n) = \dots = F_n(\alpha_1, \dots, \alpha_n) = \emptyset.$$

INPUT		
VARIABLE	DESCRIPTION	VALUE
N	order of the system	
X	maximum number of iterations	
E	tolerance	
PNLIN2()	vector of approximations (x_i)	
OUTPUT		
PNLIN2()	solution vector	
F()	vector of the values of the functions	
FUN.	Return Status	$\emptyset$ = correct calculation 1 = no solution after PNLIN2 iterations 2 = Jacobian matrix is singular 3 = algorithm diverges 4 = tolerance not positive 5 = maximum number of iterations or order not a positive integer

CALCULATION A(12)
CALLING SEQUENCE GOSUB 41001
METHOD See Appendix A
CALLED SUBROUTINES

Note

The non-linear system of equations must be included in the calling program as a subroutine whose first line is line 20. Each function is defined as $F(i)$ and each variable x_i as $PNLIN2(i)$, $i = 1, 2, \dots, N$.

For example the system:

$$y_1 = x_1^2 + x_2^2 - 4$$

$$y_2 = x_1 x_2 - 1$$

would be coded as:

```
20 F(1) = PNLIN2(1) * PNLIN2(1) + PNLIN2(2) * PNLIN2(2) - 4
```

```
30 F(2) = PNLIN2(1) * PNLIN2(2) - 1
```

```
40 RETURN
```


SOLUTION OF EQUATIONS

TEST PROGRAM FOR SLNLIN.

TEST PROGRAM NAME: NLIN										
Note: User defined system: 20(F1) = PNLIN2(1)*PNLIN2(1)+PNLIN2(2)*PNLIN2 -4 30 F(2) = PNLIN2(1)*PNLIN2(2) -1 40 RETURN										
MERGE: SLNLIN.										
INPUT	OUTPUT									
ENTER ORDER OF THE SYSTEM ? 2 ENTER TOLERANCE ? .000001 ENTER MAX. NUMBER OF ITERATIONS ? 50 APPROXIMATION VECTOR ENTER X 1 ? 0 ENTER X 2 ? 1 ENTER TOLERANCE ?	ORDER OF THE SYSTEM 2 MAX. NUMBER OF ITERATIONS 50 TOLERANCE .000001 <table> <thead> <tr> <th>i</th> <th>xi</th> <th>Fi</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>.517638068814319</td> <td>7.3476855 000365D-07</td> </tr> <tr> <td>2</td> <td>1.93185184848185</td> <td>6.0083515940 3346D-08</td> </tr> </tbody> </table>	i	xi	Fi	1	.517638068814319	7.3476855 000365D-07	2	1.93185184848185	6.0083515940 3346D-08
i	xi	Fi								
1	.517638068814319	7.3476855 000365D-07								
2	1.93185184848185	6.0083515940 3346D-08								

CC

C

C

C

CC

9. LINEAR ALGEBRA

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of problems in Linear Algebra

CONTENTS

SLCR0.	Solution of a Linear System and Matrix Inversion for a General Matrix	9-1	SLJCB.	Evaluation of the Jacobian Matrix	9-23
SLCH0.	Solution of a Linear System and Matrix Inversion for a Symmetric Positive-definite Matrix	9-5			
SLJAC.	Eigenvalues and Eigenvectors of a Symmetric Matrix using the Jacobi Method	9-10			
SLHES.	Eigenvalues of a General Matrix using the Q-R Algorithm	9-13			
SLHEV.	Eigenvectors of a General Matrix	9-17			
SLCHA.	Characteristic Polynomial of a General Matrix	9-19			

SOLUTION OF A LINEAR SYSTEM AND MATRIX INVERSION FOR A GENERAL MATRIX

SLCRO.

Solution of a linear system for multiple right-hand sides, matrix inversion and evaluation of the determinant for a general square matrix of order N

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCR01 PCR02 PCR03	order of matrix (N) tolerance(no test for singularity) switch variable	1,2 = solution of linear system -1,-2 = matrix inversion
PCR02(N,N) G(N)	elements of matrix A (overwritten by decomposition matrix) right hand side (linear system only)	
OUTPUT		
0 G(H(N)) F(N,N) FUN.	I determinant of A solution vector (linear system only) inverse matrix, A ⁻¹ (inversion only) Return Status	∅ = correct calculation 1 = order other than a positive integer 2 = tolerance ≤ ∅ or ≥ 1 3 = matrix is singular 4 = matrix is nearly singular PCR02(H(N),N) <PCR02 5 = PCR03 other than ±1 or ±2

CALCULATION	G(N) (inversion only) ; H(N)
CALLING SEQUENCE	GOSUB 41201
METHOD	See Appendix A
CALLED SUBROUTINES	

Note

PCR03 equals $\overset{+}{-}1$ for the first function call such that the matrix A is decomposed.

For a second or subsequent function call PCR03 equals $\overset{+}{-}2$ which causes the routine to bypass decomposition.

Failure to follow this procedure will generate erroneous results since the matrix A is overwritten by the decomposition matrix.

LINEAR ALGEBRA

TEST PROGRAM FOR SLCR0.

TEST PROGRAM NAME: SLCR02	
Note: In this program if the switch variable (PCR03) = 9D99 the order and the elements of the matrix can be changed	
MERGE: SLCR0.	
INPUT	OUTPUT
ENTER order of matrix ? 3 ENTER element 1,1 ? 4 ENTER element 1,2 ? 3 ENTER element 1,3 ? 9 ENTER element 2,1 ? 3 ENTER element 2,2 ? 5 ENTER element 2,3 ? 6 ENTER element 3,1 ? 4 ENTER element 3,2 ? 2 ENTER element 3,3 ? 9 ENTER tolerance ? 1d-9 ENTER 1 (system) or 2 (inverse) ? 1 ENTER b1 ? 4 ENTER b2 ? 6 ENTER b3 ? 1	matrix order 3 tolerance .000000001 right-hand side b1 = 4 b2 = 6 b3 = 1 matrix is nearly singular - last element of decomposition matrix is -.2727272727273 determinant (in absolute value) = 3.00000000001 solution x1 = -17 x2 = 3 x3 = 6.9999999998

INPUT	OUTPUT
ENTER 1 (system) or 2 (inverse) ? 2 ENTER 1 (system) or 2 (inverse) ? 9D99 ENTER ORDER OF MATRIX ?	inverse matrix -1.100000000000D+001 3.00000000000D +000 9.00000000000D+000 1.00000000000D+000 8.88178419700D- 016 -1.00000000000D+000 1.00000000000D+000 8.88178419700D -016 -1.00000000000D+000 4.6666666667D+000 -1.3333333333D+ 000 -3.6666666667D+000

SOLUTION OF A LINEAR SYSTEM AND MATRIX INVERSION FOR A SYMMETRIC POSITIVE DEFINITE MATRIX

SLCH0.

Solution of a linear system for multiple right-hand sides, matrix inversion and evaluation of the determinant for a symmetric positive definite matrix

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCH01 PCH02 PCH03 PCH02 (N(N+1)/2) F(N)	order of matrix (N) tolerance (to test for positive definiteness) switch variable elements of upper triangle of A right-hand side (linear system)	1,2 = solution of linear system -1,-2 = matrix inver- sion
OUTPUT		
F(N) 0 G(N(N+1)/2) RØ,Q FUN.	solution vector (linear system) determinant in absolute value elements of upper triangle of inverse A (inversion) see FUN. values 4 and 5 Return Status	Ø = correct calculation 1 = order other than a positive integer 2 = tolerance ≤ Ø or ≥ 1 3 = matrix not positive definite 4 = matrix not positive definite or round- ing errors. The diagonal element in row RØ of the decomposing

OUTPUT		
		matrix is not positive 5 = matrix may not be positive definite The diagonal element in row R0 = Q(<PCH02) 6 = PCH03 other than ± 1 or ± 2
CALCULATION F(N) (inversion only): A, R1, R		
CALLING SEQUENCE GOSUB 41401		
METHOD See Appendix A		
CALLED SUBROUTINES		

Note

PCH03 equals $\overset{+}{-}1$ for the first function call such that the matrix A is decomposed.

For a second or subsequent function call, PCH03 equals $\overset{+}{-}2$ which causes the routine to by-pass decomposition.

Failure to follow this procedure will generate erroneous results since the matrix A is overwritten by the decomposition matrix

TEST PROGRAM FOR SLCH0.

TEST PROGRAM NAME: SLCH02	
Note: In this program if the switch variable PCH03 = 9D99 it is possible to change the order of the matrix and the values of the elements	
MERGE: SLCH0.	
INPUT	OUTPUT
ENTER order of matrix ? 4	symmetric matrix order 4
ENTER element 1,1 ? 2	
ENTER element 1,2 ? -1	
ENTER element 1,3 ? 0	
ENTER element 1,4 ? 0	symmetric elements -1
ENTER element 2,2 ? 2	
ENTER element 2,3 ? -1	
ENTER element 2,4 ? 0	symmetric elements 0 -1
ENTER element 3,3 ? 2	
ENTER element 3,4 ? -1	symmetric elements 0 0 -1
ENTER element 4,4 ? 2	
ENTER tolerance ?.00000001	tolerance for positive definite test 00000001
ENTER 1 (system) or 2 (inverse) ? 1	right-hand side
ENTER b1 ? 1	b1 = 1
ENTER b2 ? 4	b2 = 4
ENTER b3 ? 6	b3 = 6
ENTER b4 ? 9	b4 = 9
	determinant (in absolute value) = 5.0000001601147
	solution

LINEAR ALGEBRA

INPUT	OUTPUT
ENTER 1 (system) or 2 (inverse) ? 2 ENTER 1 (system) or 2 (inverse) ? 9D99 ENTER ORDER OF MATRIX ?	X1 = 7.3999998238164 X2 = 13.7999991410504 X3 = 16.20000002010804 X4 = 12.5999996996847 Inverse matrix .800000011224664 .5999996768366 .3999999991637 .1999993595412 5999996768366 1.1999989429307 .7999997244996 .39999973499408 .39999999916397 .7999997244996 1.20000004601518 .600000003919231 .39999973499408 .19999993595412 .600000003919231 .79999976508471

EIGEN VALUES AND EIGEN VECTORS OF A SYMMETRIC MATRIX USING THE JACOBI METHOD

SLJAC.

Calculation of the eigen values and, optionally, the eigen vectors of a symmetric matrix A, i.e. solution of the matrix equation $Ax = \lambda x$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PJAC1 PJAC2 PJAC3 PJAC2(N) F(N(N-1)/2)	order of the matrix (N) tolerance switch variable diagonal elements of the matrix upper off-diagonal elements of the matrix	$\emptyset$ = without eigen vectors 1 = with eigen vectors
OUTPUT		
PJAC2(N) VJAC7(N,N) FUN.	eigen values eigen vectors (columns of VJAC7() Return Status 	

CALCULATION	F, B, J, T_1 , T_2 , U, $C\emptyset$, $S\emptyset$, D, Q, R
CALLING SEQUENCE	GOSUB 41601
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLJAC.

TEST PROGRAM NAME: SLJAC2	
Note:	
MERGE: SLJAC.	
INPUT	OUTPUT
ENTER order of matrix ? 3 ENTER element 1,1 ? 8 ENTER element 1,2 ? 4 ENTER element 1,3 ? 2 ENTER element 2,2 ? 6 ENTER element 2,3 ? 3 ENTER element 3,3 ? 9 ENTER tolerance ? 1d-10 eigen vectors ? 1(yes), 0(no) ? 1 ENTER order of matrix ?	symmetric matrix order 3 symmetric elements 4 symmetric elements 2 3 tolerance .00000000001 eigen vectors requested eigen values 13.687339256 2.619921695 6.692739049 eigen vectors (row-wise) .590940671 .541016969 .598406026 -.527975155 .820223273 -.220172769 -.609943755 -.185834468 .770346785

EIGEN VALUES OF A GENERAL MATRIX USING THE Q-R ALGORITHM

SLHES.

Calculation of the eigen values of a general matrix A, i.e. solution of the matrix equation $Ax = \lambda x$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PHES1 PHES2 E(N,N)	order of the matrix (N) tolerance elements of the matrix	
OUTPUT		
F(N),G(N) E(N,N) P(N,N) FUN.	real and imaginary parts of the eigen values matrix U matrix W=PQ } see method Return Status	$\emptyset$ = eigen values calculated 1 = PHES1 is a non positive integer 2 = PHES2 is not positive 3 = no convergence after 3 $\emptyset$ iterations

CALCULATION	O(N), L!, K!, B, Q, R, S, X, Y, W, Z
CALLING SEQUENCE	GOSUB 41801
METHOD	See Appendix A
CALLED SUBROUTINES	GOSUB 46001

Note

The progress of the calculation may be followed on the display by removing REM from line 41893. The display then shows the iteration being performed in each of the N-2 steps of the Q-R algorithm

LINEAR ALGEBRA

TEST PROGRAM FOR SLHES. and SLHEV.

TEST PROGRAM NAME: SLHES2	
Note: The calculation of the eigen values is followed by the calculation of the corresponding eigen vectors using SLHEV. (GOSUB 46001)	
MERGE: SLHES. SLHEV.	
INPUT	OUTPUT
ENTER order ? 4 ENTER elements row-wise ENTER element row 1 a1 1 = ? 1 a1 2 = ? 5 a1 3 = ? 8 a1 4 = ? 4 ENTER element row 2 a2 1 = ? 6 a2 2 = ? 3 a2 3 = ? 7 a2 4 = ? 8 ENTER element row 3 a3 1 = ? 1.5 a3 2 = ? 11 a3 3 = ? 9 a3 4 = ? 6 ENTER element row 4 a4 1 = ? 5 a4 2 = ? 8 a4 3 = ? 3.7 a4 4 = ? 4	MATRIX (row-wise) 1 5 8 4 6 3 7 8 1.5 11 9 6 5 8 3.7 4

INPUT	OUTPUT
tolerance ? 1d -10	j eigen value j 1 -5.12610607584948 2 -.579869218693827+2.42693 114280701i 3 -.5798621893827+-.242 693114280701 4 23.2858438460084 j eigen vector j 1 -.231220286728237 -.6932206133 52799 .304962295655581 .610721181895829 2 and 3 -.545794465208018+-.2210- 85637759322.1 0370383056315 +-1.48041 119486146i -.327278951080694+-.35815 5113472947i .610288157833865+-.112785 796835097i 4 -.305850574192517 -.560817326238133 -.595621533278672 -.487005517499808 ENTER order ?

EIGEN VECTORS OF A GENERAL MATRIX

Calculation of the eigen vectors of a general matrix A, i.e. solution of the matrix equation $Ax = \lambda x$, given the eigen values λ_i , and the matrices U and W, where $U = Q^t P^t A P Q$ and $W = P Q$ (see SLHES.)

INPUT		
VARIABLE	DESCRIPTION	VALUE
PHLEV1 PHLEV2 E(N,N) P(N,N) F(N),G(N)	order of matrix A (N) tolerance matrix U matrix W real and imaginary parts of eigen values of A	} see method
OUTPUT		
P(N,N) FUN.	eigen vectors of A (see note) Return Status	$\emptyset$ = correct calculation 1 = tolerance not positive 2 = order other than a positive integer

CALCULATION	O(N), B, H, M, P, Q, R1, S1, T1, T2, Z, Z1, Z2, Y2
CALLING SEQUENCE	GOSUB 46001
METHOD	See Appendix A
CALLED SUBROUTINES	

Notes

1. If λ_i is i^{th} eigen value and x_{ij} the j^{th} element of the i^{th} eigen vector then for λ_i real, $P(i,j) = x_{ij}$. If λ_i is complex ($G(i) \neq \emptyset$) then λ_{i+1} is its conjugate; $P(i,j)$ is then the real part of x_{ij} and $x_{i+1,j}$ and $P(i+1,j)$ and $-P(i+1,j)$ are the imaginary parts of x_{ij} and $x_{i+1,j}$ respectively
2. For the test program see SLHES.

CHARACTERISTIC POLYNOMIAL OF A GENERAL MATRIX

SLCHA.

Calculation of the coefficients of the characteristic polynomial of a general matrix A. Evaluation of the determinant (in absolute value)

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCHA1 E(N,N)	order of the matrix (N) elements of the matrix	
OUTPUT		
H(N+1) FUN.	Coefficients of the characteristic polynomial in descending order. (The absolute value of the determinant of A is the absolute value of H (N+1)). Return Status	$\emptyset$ = correct calculation 1 = order N is not a positive integer

CALCULATION	$G(N)$, $F(N,N)$
CALLING SEQUENCE	GOSUB 422Ø1
METHOD	See Appendix A
CALLED SUBROUTINES	

LINEAR ALGEBRA

TEST PROGRAM FOR SLCHA.

TEST PROGRAM NAME: SLCHA2																	
Note:																	
MERGE: SLCHA.																	
INPUT	OUTPUT																
ENTER ORDER OF MATRIX ? 4 ENTER elements row 1 E(1,1) ? 7 E(1,2) ? 6 E(1,3) ? 5 E(1,4) ? 3 ENTER elements row 2 E(2,1) ? 2 E(2,2) ? 8 E(2,3) ? 9 E(2,4) ? 11 ENTER elements row 3 E(3,1) ? 5 E(3,2) ? 4 E(3,3) ? 8 E(3,4) ? 6 ENTER elements row 4 E(4,1) ? 3 E(4,2) ? 5 E(4,3) ? 7 E(4,4) ? 2	MATRIX (row-wise) <table> <tr> <td>7</td> <td>6</td> <td>5</td> <td>3</td> </tr> <tr> <td>2</td> <td>8</td> <td>9</td> <td>11</td> </tr> <tr> <td>5</td> <td>4</td> <td>8</td> <td>6</td> </tr> <tr> <td>3</td> <td>5</td> <td>7</td> <td>2</td> </tr> </table>	7	6	5	3	2	8	9	11	5	4	8	6	3	5	7	2
7	6	5	3														
2	8	9	11														
5	4	8	6														
3	5	7	2														

INPUT	OUTPUT
ENTER ORDER OF MATRIX?	COEFFICIENTS OF THE POLYNOMIAL (in descending order) 1 -25 43 182 -1040

EVALUATION OF THE JACOBIAN MATRIX

To evaluate the Jacobian matrix of a user defined system of functions

INPUT		
VARIABLE	DESCRIPTION	VALUE
PJCB1 PJCB2 F(N) G(N)	order of the system (N) increment for the five point differencing scheme functions variables	
OUTPUT		
E(N,N) FUN.	Jacobian matrix Return Status	$\emptyset$ = correct calculation 1 = order of system is not a positive integer 2 = increment is not positive

CALCULATION	B()
CALLING SEQUENCE	GOSUB 42401
METHOD	See Appendix A
CALLED SUBROUTINES	

Note

The system of equations must be included in the calling program as a subroutine whose first line must be line 20.

For example, the system

$$\begin{aligned}y_1 &= x_1 - x_2 \\ y_2 &= x_1 x_2\end{aligned}$$

would be coded in the calling program as follows:

20 F(1) = G(1) - G(2)

30 F(2) = G(1) * G(2)

40 RETURN

LINEAR ALGEBRA

TEST PROGRAM FOR SLJCB.

TEST PROGRAM NAME: SLJCB2	
Note: user defined functions: 20 F(1) = G(1) - G(2) 30 F(2) = G(1) * G(3) 40 RETURN	
MERGE: SLJCB.	
INPUT	OUTPUT
ENTER order of system ? 2 ENTER increment ? .01 ENTER X 1 ? 6 ENTER X 2 ? 3 ENTER increment ? 0 ENTER X 1 ? 1 ENTER X 2 ? 1 ENTER increment ?	increment .01 X1 = 6 X2 = 3 Jacobian matrix .9999999999999979 -1.0000000000000000 0001 3.00000000000014 6.000000000000005 increment 0 X1 = 1 X2 = 1 error positive increment only

”

”

”

”

”

10. CURVE FITTING AND INTERPOLATION

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of problems in Curve Fitting and Interpolation.

CONTENTS

SLLSQ.	Least Squares Curve Fitting to User Supplied Basis	10-1	SLCSPI.	Cubic Spline Interpolation	10-24
SLNLLS.	Non-linear Least Squares Curve Fitting to an Arbitrary Scalar Function	10-4	SLPADE.	Rational Function Fitting (Padè Approximation)	10-28
SLFSYT.	Weighted Least Squares Orthogonal Polynomial Curve Fit (Forsythe)	10-8			
SLFTRP.	Coefficients of Fourier Series to Represent Discrete Data	10-12			
SLFOUI.	Fourier Interpolation	10-17			
SLPLYF.	Lagrangian Interpolation	10-19			
SLCSPC.	Cubic Spline	10-23			

CURVE FITTING AND INTERPOLATION

LEAST SQUARES CURVE FITTING TO USER SUPPLIED BASIS

SLLLSQ.

To determine the coefficients of an approximating function, which is a linear combination of user-supplied basis functions

INPUT		
VARIABLE	DESCRIPTION	VALUE
PLLSQ1 PLLSQ2 PLLSQ3 E(N1),F(N1)	number of data points (N1) number of basis functions (M1) tolerance x_i, y_i	$i = 1, 2, \dots, N1$
OUTPUT		
G(M1) D N! FUN.	basis coefficients standard deviation number of basis functions used Return Status	$\emptyset$ = tolerance reached 1 = tolerance not reached 2 = number of data points less than the number of functions 3 = normal system is singular 4 = tolerance negative

OUTPUT		
		5 = number of data points other than a positive integer 6 = number of basis functions other than 1,2...6
CALCULATION $Q(,)$, $R(,)$, $H()$, $O()$, P , U , F		
CALLING SEQUENCE GOSUB 42601		
METHOD See Appendix A		
CALLED SUBROUTINES		

Note

The basis functions must be included in the calling program as a subroutine whose first line must be line 20. For example, the basis

$$\begin{aligned} f_1 &= x \\ f_2 &= x^2 \end{aligned}$$

would be coded in the calling program as:

```
20 G(1) = x
30 G(2) = x↑2
40 RETURN
```

CURVE FITTING AND INTERPOLATION

TEST PROGRAM FOR SLLSQ.

TEST PROGRAM NAME: RRLLSQ															
Note: user-defined basis functions: 20 G(1) = X 30 G(2) = X^2 40 RETURN															
MERGE: SLLSQ.															
INPUT	OUTPUT														
ENTER number of points ? 7 ENTER interval limits ? 1.1, 9 ENTER E(1) F(1) ? 1.1, 3 ENTER E(2) F(2) ? 2.3, 0 ENTER E(3) F(3) ? 3, 4 ENTER E(4) F(4) ? 4.5, -3 ENTER E(5) F(5) ? 6, -12 ENTER E(6) F(6) ? 9, 456 ENTER E(7) F(7) ? 12, 4 ENTER number of basis functions ? 6 ENTER tolerance ? 0 ENTER tolerance ?	number of points = 7 interval [1.1, 9] 6 basis functions tolerance = 0 ***results*** 6 basis functions used <table border="1"> <thead> <tr> <th>basis fn</th> <th>coefficient</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>207.413547964336</td> </tr> <tr> <td>2</td> <td>-362.077179199238</td> </tr> <tr> <td>3</td> <td>213.013346355761</td> </tr> <tr> <td>4</td> <td>-53.9416220909832</td> </tr> <tr> <td>5</td> <td>5.94626574262395</td> </tr> <tr> <td>6</td> <td>-.227555306780735</td> </tr> </tbody> </table> standard deviation = 2.21041178703308 (tolerance not reached)	basis fn	coefficient	1	207.413547964336	2	-362.077179199238	3	213.013346355761	4	-53.9416220909832	5	5.94626574262395	6	-.227555306780735
basis fn	coefficient														
1	207.413547964336														
2	-362.077179199238														
3	213.013346355761														
4	-53.9416220909832														
5	5.94626574262395														
6	-.227555306780735														

NON-LINEAR LEAST SQUARES CURVE FITTING TO AN ARBITRARY SCALAR FUNCTION

SLNLLS.

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PNLLS1	number of data points (N)	≤ 6
PNLLS2	number of parameters in model	
PNLLS3	tolerance to determine presence of maximum	
PNLLS4	tolerance for line minimisation	
PNLLS5	bound for line minimisation	
PNLLS6	maximum number of iterations	
P(PNLLS2)	initial approximation to a set of parameters	
x(PNLLS1), y(PNLLS1)	x and y data values	
OUTPUT		
P(PNLLS2)	final set of parameters	$\emptyset$ = correct calculation 1 = no solution after PNLLS6 iterations 2 = max. no. of iterations not a positive integer 3 = tolerances and bound not positive 4 = no. of data points not a positive integer 5 = no. of parameters other than 1,2,...,6
F	residual value	
K1	number of iterations	
FUN.	Return Status	

CURVE FITTING AND INTERPOLATION

CALCULATION	S(,),D(),Q(),Z(),B0,H,I!,J!,F9,M1,M2,S,Q,Z,T1
CALLING SEQUENCE	GOSUB 46401
METHOD	See Appendix A
CALLED SUBROUTINES	

Notes

1. The model $f(a_1, a_2, \dots, a_k, x)$ must be defined in the calling program as FNF(X), with argument X and parameters P(1), P(2), ..., P(6). For example, the modified exponential curve $a_1 + a_2 \cdot a_3^x$ would be coded as DEF FNF(X) = P(1) + P(2) * P(3) $\uparrow$ X
2. The progress of calculation may be followed on the display by removing REM from statements 46472 and 46655. The last line minimisation and value of the residual are given

TEST PROGRAM FOR SLNLLS.

TEST PROGRAM NAME: RRNLLS	
Note: user-defined model $3\emptyset \text{ DEF FNF}(x) = P(1) + P(2) * P(3) \hat{x}$	
MERGE: SLNLLS.	
INPUT	OUTPUT
ENTER number of points ? 24 ENTER interval limits ? .2, 7.3 X(1) Y(1) ? .2, 24.3 X(2) Y(2) ? .5, 23.5 X(3) Y(3) ? .7, 22.8 X(4) Y(4) ? .8, 22.5 X(5) Y(5) ? 1.2, 21.5 X(6) Y(6) ? 1.4, 21.1 X(7) Y(7) ? 1.7, 20.5 X(8) Y(8) ? 1.9, 20.1 X(9) Y(9) ? 2.3, 19.4 X(10) Y(10) ? 2.5, 19.1 X(11) Y(11) ? 2.7, 18.8 X(12) Y(12) ? 3, 18.4 X(13) Y(13) ? 3.3, 18.1 X(14) Y(14) ? 3.6, 17.8 X(15) Y(15) ? 3.9, 17.5 X(16) Y(16) ? 4.1, 17.3 X(17) Y(17) ? 4.4, 17.1 X(18) Y(18) ? 4.7, 16.9 X(19) Y(19) ? 3, 16.7 X(20) Y(20) ? 5.3, 16.3 X(21) Y(21) ? 5.7, 16.3 X(22) Y(22) ? 6.2, 16.1 X(23) Y(23) ? 6.7, 15.9	number of points 24 interval [.2, 7.3]

CURVE FITTING AND INTERPOLATION

INPUT	OUTPUT								
X(24) Y(24) ? 7.3, 15.7 ENTER number of parameters ? 3 ENTER tolerance ? .001 ENTER line min. tolerance ? .001 ENTER bound for line min. ? 5 ENTER maximum # of iterations? 50 initial parameters: ENTER parameter1 ? 10 ENTER parameter2 ? 10 ENTER parameter3 ? 1 ENTER tolerance ?	number of parameters 3 bound for line min. 5 solution after 34 iterations <table> <tr> <th>i</th><th>parameter i</th></tr> <tr> <td>1</td><td>15.1539120999214</td></tr> <tr> <td>2</td><td>9.94726076961454</td></tr> <tr> <td>3</td><td>.683799252607205</td></tr> </table> residual value = .932332519366355	i	parameter i	1	15.1539120999214	2	9.94726076961454	3	.683799252607205
i	parameter i								
1	15.1539120999214								
2	9.94726076961454								
3	.683799252607205								

WEIGHTED LEAST SQUARED ORTHOGNAL POLYNOMIALS CURVE FIT (FORSYTHE)

SLFSYT.

To compute the coefficients of the Forsythe polynomials approximating
a set of observed data

INPUT		
VARIABLE	DESCRIPTION	VALUE
PFSYT1 PFSYT2 PFSYT3 E(N) F(N) G(N)	tolerance maximum degree of resultant poly- nomial (M) number of points (N) array of data points abscissae array of data points ordinates array of data points weights	
OUTPUT		
P1 R(P1+1) SØ FUN.	degree of resultant polynomial coefficients of polynomial in ascending order standard deviation Return Status	Ø = correct calculation 1 = tolerance not reached 2 = no. of data points other than a posi- tive integer 3 = maximum degree of polynomial other than Ø,1...N-1 4 = tolerance negative

CURVE FITTING AND INTERPOLATION

CALCULATION	H (7, M+2), D1
CALLING SEQUENCE	GOSUB 43001
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLFSYT.

TEST PROGRAM NAME: RRFST	
Note:	
MERGE: SLFSYT.	
INPUT	OUTPUT
ENTER number of points ? 24 ENTER interval limits ? .2, 7.3 E(1) F(1) G(1) ? .2,24.3,1 E(2) F(2) G(2) ? .5,23.4,1 E(3) F(3) G(3) ? .7,22.8,1 E(4) F(4) G(4) ? .8,22.5,1 E(5) F(5) G(5) ? 1.2,21.5,1 E(6) F(6) G(6) ? 1.4,21.1,1 E(7) F(7) G(7) ? 1.7,20.5,1 E(8) F(8) G(8) ? 1.9,20.1,1 E(9) F(9) G(9) ? 2.3,19.4,1 E(10) F(10) G(10) ? 2.5,19.1,1 E(11) F(11) G(11) ? 2.7,18.8,1 E(12) F(12) G(12) ? 3,18.4,1 E(13) F(13) G(13) ? 3.3,18.1,1 E(14) F(14) G(14) ? 3.6,17.8,1 E(15) F(15) G(15) ? 3.9,17.5,1 E(16) F(16) G(16) ? 4.1,17.3,1 E(17) F(17) G(17) ? 4.4,17.1,1 E(18) F(18) G(18) ? 4.7,6.9,1 E(19) F(19) G(19) ? 5 ,16.7,1 E(20) F(20) G(20) ? 5.3,16.5,1 E(21) F(21) G(21) ? 5.7,16.3,1 E(22) F(22) G(22) ? 6.2,16.1,1	number of points = 24 interval [.2, 7.3]

CURVE FITTING AND INTERPOLATION

INPUT	OUTPUT
E(23) F(23) G(23) ? 6.7,15.9,1 E(24) F(24) G(24) ? 7.3,15.7,1 ENTER x,y scale factors ? 1,1 ENTER tolerance ? 0 ENTER maximum degree ? 10 ENTER tolerance ?	scale factor x = 1 y = 1 tolerance = 0 maximum degree = 10 degree of polynomial is 5 coefficients (in descending order) - 5.40069907256268D-06 1.72434804771512D-03 -47389369009 4188D-02 .558221064403478 -3.475- 68158279705 24.9769520508887 standard deviation = 1.91369075328 112D-02

COEFFICIENTS OF FOURIER SERIES TO REPRESENT DISCRETE DATA

SLFTRP.

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PFTRP1 PFTRP2 PFTRP3 X(N),F(N)	maximum harmonic order to be considered tolerance number of base points (N) N base points and function values	≥ 0
OUTPUT		
P1! G(P1!), H(P1!) E(P1!) P FUN.	number of harmonics computed coefficients a_k and b_k least squares error at step k; if N is even and $P1! = M+2$ then a dummy value of 9D99 is returned for E(M+1) see FUN. value 1 Return Status	$K = 0, 1, \dots, P1!-1$ 0 = correct calculation 1 = maximum harmonic order other than $\frac{n}{2}$ $0, 1, \dots, P$ where $P = \frac{n}{2}$ for N even, $P = \frac{N-1}{2}$ for N odd 3 = tolerance negative

CURVE FITTING AND INTERPOLATION

OUTPUT		
		4 = number of base points other than a positive integer
CALLING SEQUENCE GOSUB 43201		
METHOD See Appendix A		
CALLED SUBROUTINES		

TEST PROGRAM FOR SLFTRP. and SLFOUI.

[illegible]

CURVE FITTING AND INTERPOLATION

INPUT	OUTPUT
ENTER max harmonic order ? 10	maximum harmonic order 10 k 0 ak=0 bk=0 error at order 0 170 1 ak=3.00000002084744D-07 bk=4.086343 error at order 1 3.018 2 ak=-5.57295000386778D-07 bk=.000000764120805 error at order 2 3.018 3 ak=-6.00000000172685D-07 bk=.4851837 error at order 4 .664 4 ak=3.21345966410038D-16 bk=1.68941605D-16 error at order 4 .664 5 ak=2.00000000280431D-08 bk=.2 error at order 5 .264 6 ak=6.07393800153443D-08 bk=000000158873905 error at order 6 .264 7 ak=4.19999999978771D-07 bk=.1259615 error at order 7 .1053 8 ak=0 bk=0 error at order 8 .1053 9 ak=-5.100000000009114D-07 bk=.1025091 error at order 9 .0002000000005 10 ak=0 bk=.0000001 error at order 10 ***** interpolation <u>scale codes</u> 1 original 2 unit spacing 3 radians 4 degrees
SELECT CODE ? 1	

INPUT	OUTPUT
ENTER X ? 2 ENTER X ? 3 ENTER X ? 4 ENTER X ? 9D99 SELECT CODE ? 2 ENTER X ? 2 ENTER X ? 9D99 SELECT CODE ? 3 ENTER X ? 1.5707963 ENTER X ? 9D99 SELECT CODE ?	$x=2 \quad y(x)=.999999070212393 \quad \text{Code1}$ $x=3 \quad y(x)=1.44573944215214 \quad \text{Code1}$ $x=4 \quad y(x)=1.99999698837309 \quad \text{Code1}$ $x=2 \quad y(x)=1.99999698837309 \quad \text{Code2}$ $x=1.5707963$ $y(x)=4.99999779655506 \quad \text{Code}$

FOURIER INTERPOLATION

Evaluation of the Fourier approximating expansion

INPUT		
VARIABLE	DESCRIPTION	VALUE
PFOUI1	scale code	{ 1 = original 2 = unit spacing 3 = radians 4 = degrees
PFOUI2	abscissa of first observation	
PFOUI3	increment between two successive abscissae (H)	
PFOUI4	abscissa to be interpolated	
PFOUI5	number of observations (N)	
PFOUI6	maximum harmonic order of Fourier expansion (M)	
G(M+1), H(M+1)	Fourier Coefficients for cosines and sines respectively	
OUTPUT		
C FUN.	value of Fourier expansion Return Status	Ø = correct calculation 1 = scale code other than 1,2,3 or 4 2 = no. of observations other than a positive integer 3 = maximum harmonic order other than Ø,1...P, where $P = \frac{n}{2}$ for N even, $P = \frac{n-1}{2}$ for N odd

CALLING SEQUENCE GOSUB 43401
METHOD See Appendix A
CALLED SUBROUTINES

Note

For the test program for this subroutine see SLFTRP.

LAGRANGIAN INTERPOLATION

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

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPLYF1 X(N),Y(N)	number of points (N) given x,y values	
OUTPUT		
E(N) FUN.	coefficients of interpolating polynomial in ascending order Return Status	$\emptyset$ = correct calculation 1 = two x values equal. Denominator A_i equals zero (see Method) 2 = number of points other than a posi- tive integer

CALCULATION	H(N)
CALLING SEQUENCE	GOSUB 436Ø1
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLPLYF.

Note:

INPUT

OUTPUT

```
number of points = 11
interval [-5, 5]
```

X(1)	Y(1)	?	-5,19900
X(2)	Y(2)	?	-4,-525
X(3)	Y(3)	?	-3,-96
X(4)	Y(4)	?	-2,-15
X(5)	Y(5)	?	-1,0
X(6)	Y(6)	?	0,15
X(7)	Y(7)	?	1,0
X(8)	Y(8)	?	2,-51
X(9)	Y(9)	?	3,0
X(10)	Y(10)	?	4,555
X(11)	Y(11)	?	5,2400

coefficients of polynomial

order	coefficient
10	-1.0842021724855D-19
9	4.33680868994202D-19
8	0
7	1.38777878078145D-17
6	-2.77555756156289D-17
5	1
4	1

INPUT	OUTPUT								
ENTER X ($-5 \leq X \leq 5$) ? 2,5 ENTER X ($-5 \leq X \leq 5$) ? 3.1 ENTER X ($-5 \leq X \leq 5$) ? -1.5 ENTER X ($-5 \leq X \leq 5$) ? 4.7 ENTER X ($-5 \leq X \leq 5$) ?	<table> <tr> <td>3</td><td>-8</td></tr> <tr> <td>2</td><td>-16</td></tr> <tr> <td>1</td><td>7</td></tr> <tr> <td>Ø</td><td>15</td></tr> </table> <u>interpolation</u> X=2.5 Y(X)=55.78125 X=3.1 Y(X)=23.25561Ø X=1.5 Y(X)=7.Ø3125 X=4.7 Y(X)=1645.2942	3	-8	2	-16	1	7	Ø	15
3	-8								
2	-16								
1	7								
Ø	15								

CUBIC SPLINE

To determine the interpolating cubic spline for the given data set

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCSPC1 PCSPC2 PCSPC3 F(N),G(N)	number of data points (N) 2nd derivative at the first point 2nd derivative at the last point x,y data values	
OUTPUT		
E(N-1,4) FUN.	matrix of coefficients of cubic spline (i.e. $E(1,1) = a_3 \dots$ $E(1,4) = a_0$) Return Status	See Method $\emptyset$ = correct calculation 1 = data not increasing order of x 2 = number of data points other than a positive integer
CALLING SEQUENCE GOSUB 438 $\emptyset$ 1		
METHOD See Appendix A		
CALLED SUBROUTINES		

CUBIC SPLINE INTERPOLATION

Given a data set $x_i, y_i, i = 1, 2, \dots, N$ and a set of interpolatory cubics $f_i(x), i = 1, 2, \dots, N-1$, to evaluate the cubic spline for $x_1 \leq x \leq x_n$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PCSPI1 PCSPI2 F(N),G(N) E(N-1,4)	number of data points (N) value to be interpolated x_i, y_i data values coefficients of interpolatory cubics in descending order (i.e. $E(I,1) = a_3 \dots E(I,4) = a_0$)	$i = 1, \dots, N$
OUTPUT		
P FUN.	value of cubic spline at x Return Status	$\emptyset$ = correct calculation 1 = data not in increasing order of x 2 = number of data points other than a positive integer 3 = x outside the range of given x values
CALLING SEQUENCE GOSUB 44001		
METHOD See Appendix A		
CALLED SUBROUTINES		

CURVE FITTING AND INTERPOLATION

Note

For the test program for this routine see SLCSPC

TEST PROGRAM FOR SLCSPC. and SLCSPI.

TEST PROGRAM NAME: RRSPC																															
Note: The test program uses SLCSPI. to evaluate the cubic spline. In this program if the i value to be interpolated (PCSPI2) is equal to 9D99 the data values (xi,yi) can be changed																															
MERGE: SLCSPC. SLCSPI.																															
INPUT	OUTPUT																														
ENTER number of points ? 8 ENTER interval limits ? 0, 7 F(1) G(1) ? 0,0 F(2) G(2) ? 1,1 F(3) G(3) ? 2,4 F(4) G(4) ? 3,9 F(5) G(5) ? 4,16 F(6) G(6) ? 5,25 F(7) G(7) ? 6,36 F(8) G(8) ? 7,49 ENTER $y''(x_1), y''(x_n)$? 2,2	$y''(x_1) = 2$ $y''(x_n) = 2$ spline coefficients <table> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td></td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td></td><td>2</td><td>1</td></tr> <tr><td>2</td><td>0</td><td>1</td></tr> <tr><td></td><td>4</td><td>4</td></tr> <tr><td>3</td><td>0</td><td>1</td></tr> <tr><td></td><td>6</td><td>19</td></tr> <tr><td>4</td><td>0</td><td>1</td></tr> <tr><td></td><td>8</td><td>16</td></tr> </table>	0	0	1		0	0	1	0	1		2	1	2	0	1		4	4	3	0	1		6	19	4	0	1		8	16
0	0	1																													
	0	0																													
1	0	1																													
	2	1																													
2	0	1																													
	4	4																													
3	0	1																													
	6	19																													
4	0	1																													
	8	16																													

CURVE FITTING AND INTERPOLATION

INPUT	OUTPUT
ENTER X ? 1 ENTER X ? 2.5 ENTER X ? 3.35 ENTER X ? 6.15 ENTER X ? 9D99 ENTER number of points	5 0 1 10 25 6 0 1 12 36 <u>interpolation</u> X=1 Y=1 X=2.5 Y=6.25 X=3.35 Y=11.2225 X=6.15 Y=37.8225

**RATIONAL FUNCTION FITTING
(PADE' APPROXIMATION)**

SLPADE.

To calculation the coefficients of the rational function (a quotient of two polynomials of specified degree) which best approximates a given function on an interval

INPUT		
VARIABLE	DESCRIPTION	VALUE
PPADE1	degree of the numerator (M)	<10
PPADE2	degree of the denominator (N)	<10
PPADE3	to enter Taylor coefficients set PPADE3 = 1 to enter derivatives set PPADE3 = 2	
E(M,N+1)	coefficients of Taylor series or derivatives	
OUTPUT		
F(M+1)	coefficients of numerator	$\emptyset$ = correct calculation 1 = impossible to solve the system 2 = degree of numerator or denominator other than a positive integer <10 3 = PPADE3 other than 1 or 2
G(N+1)	coefficients of denominator	
FUN.	Return Status	

CURVE FITTING AND INTERPOLATION

CALCULATION	$H(N)$, $O(N+1)$, $P(N,M+1)$
CALLING SEQUENCE	GOSUB 442Ø1
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLPAD.

TEST PROGRAM NAME: RRPAD															
Note: In this program if the argument x or the centre point = 9D99 it is possible to delete the f(x) function and the sum of the degrees of the numerator and the denominator must be < 10															
MERGE: SLPAD. SLPLRR.															
INPUT	OUTPUT														
ENTER 1 (coeffs) or 2 (derivs) ? 2 ENTER degree numerator ? 2 ENTER degree denominator ? 3 ENTER derivative of order 0 ? 1 ENTER derivative of order 1 ? 1 ENTER derivative of order 2 ? 1 ENTER derivative of order 3 ? 1 ENTER derivative of order 4 ? 1 ENTER derivative of order 5 ? 1 ENTER centre point ? 0	degree numerator = 2 degree denominator = 3 <table> <tr> <th>order</th> <th>derivative</th> </tr> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>1</td> </tr> <tr> <td>3</td> <td>1</td> </tr> <tr> <td>4</td> <td>1</td> </tr> <tr> <td>5</td> <td>1</td> </tr> </table> degree of the numerator is 2 coefficients in descending order .05 .4 1 degree of the denominator is 3 coefficients in descending order -1.6666666666666666D-02 .15 -.6 1 evaluation	order	derivative	0	1	1	1	2	1	3	1	4	1	5	1
order	derivative														
0	1														
1	1														
2	1														
3	1														
4	1														
5	1														

CURVE FITTING AND INTERPOLATION

INPUT	OUTPUT
ENTER X ? 2.1 ENTER X ? 0	$X=2.1 \quad f(x)=8.33704228201497$ $X=0 \quad f(x)=1$

”

”

”

”

”

11. NUMERICAL INTEGRATION AND DIFFERENTIATION

ABOUT THIS CHAPTER

This chapter contains subroutines for the solution of problems in Numerical Integration and Differentiation.

CONTENTS

SLITEQ.	Integration of a Tabulated Function (Equal Spacing)	11-1
SLDTEQ.	Differentiation of a Tabulated Func- tion (Equal Spa- cing)	11-5
SLDIFF.	Differentiation of a Non-tabulated Function	11-9
SLROMB.	Romberg Integration	11-13
SLGAUS.	Guass - Legendre Quadrature	11-16
SLLAGU.	Guass - Laguerre Quadrature	11-19

INTEGRATION OF A TABULATED FUNCTION (EQUAL SPACING)

SLITEQ.

Integration of a tabulated function for equally spaced abscissae
 (x_i, y_i) $i = 1, \dots, N$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PITEQ1 PITEQ2 X(N+1), Y(N+1)	number (N) of equally spaced abscissae minus 1; i.e. the number of subdivisions in the interval. N must be a positive integer $\geq P$ (order of quadrature) order of quadrature $P = 1, 2 \dots 8$ but if $N < 8$ then $P \leq N$ the N+1 abscissae and ordinates of the tabulated function (in ascending order of X)	
OUTPUT		
I1 R FUN.	value of the integral order of quadrature on remainder Return Status	$\emptyset$ = correct calculation 1 = order of quadrature other than 1, 2, ..., 8 2 = invalid number of subdivisions 3 = data not equally spaced 4 = start interval $\geq$ end

CALCULATION	PITEQ2(P+1); stores the coefficients of the quadrature of the formula
CALLING SEQUENCE	GOSUB 44401
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLITEQ.

11-3

INPUT	OUTPUT
ENTER ORDER OF QUADRATURE ?	3 SUBDIVISIONS VALUE OF INTEGRAL=41.066667

NUMERICAL INTEGRATION AND DIFFERENTIATION

DIFFERENTIATION OF A TABULATED FUNCTION (EQUAL SPACING)

SLDTEQ.

Differentiation of a tabulated function for equally spaced abscissae
 $(x_i, y_i) \ i = 1, \dots, N$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PDTEQ1	number (N) of equally spaced abscissae minus 1 i.e. the number of subdivisions in the interval. N must be an integer ≥ 5	1 = first derivative only 2 = first and second derivatives
PDTEQ2	derivative evaluations	
X(N+1), Y(N+1)	the N+1 abscissae and ordinates of the tabulated function	
OUTPUT		
S(N+1), Q(N+1) FUN.	first and second derivative at each of the N+1 base points Q() is returned only for PDTEQ2 = 2 Return Status	$\emptyset$ = correct calculation 1 = PDTEQ2 other than 1 or 2 2 = no. of subdivisions not an integer ≥ 5 3 = data not equally spaced 4 = start interval $\geq$ end

CALCULATION	A(), B() store the coefficients of the difference formula
CALLING SEQUENCE	GOSUB 446Ø1
METHOD	See Appendix A
CALLED SUBROUTINES	

TEST PROGRAM FOR SLDTEQ.

11-7

INPUT	OUTPUT				
ENTER1 (Y') OR 2 (Y'&Y'') ? 2	10	4.5	20.25	9	
	11	5	25	10	
ENTER1 (Y') OR 2 (Y'&Y'') ?	i	Xi	Yi	Yi'	Yi''
	1	0	0	0	2
	2	.5	.25	1	2
	3	1	1	2	2
	4	1.5	2.25	3	2
	5	2	4	4	2
	6	2.5	6.25	5	2
	7	3	9	6	2
	8	3.5	12.25	7	2
	9	4	16	8	2
	10	4.5	20.25	9	2
	11	5	25	10	2

DIFFERENTIATION OF A NON-TABULATED FUNCTION

SLDIFF.

Evaluation of the first and second derivatives of a user-defined function given in non-tabular form

INPUT		
VARIABLE	DESCRIPTION	VALUE
PDIFF1	point at which derivatives are to be evaluated	1 = evaluate first derivative only 2 = evaluate first and second derivatives
PDIFF2	increment (H) used in 5-point central difference formula H must be positive	
PDIFF3	derivative switch	
OUTPUT		
D1,D2	first and second derivatives of the function at PDIFF1. D2 is returned only for PDIFF3 = 2	Ø = correct calculation 1 = increment non-positive 2 = PDIFF3 other than 1 or 2
FUN.	Return Status	

CALLING SEQUENCE	GOSUB 44801
METHOD	See Appendix A
CALLED SUBROUTINES	

Notes

1. The function to be differentiated must be included in the calling program as FNF.

For example:

```
20 DEF FNF(X) = X * X
```

would describe the function $f(x) = x^2$

2. The increment H is set by the user and the following rules will give accurate results:

$h \geq 0.005$ to evaluate the first derivative only

$h \geq 0.06$ (minimum) to evaluate both derivatives

INPUT	OUTPUT			
ENTER INCREMENT ?	6.00	36.00	12.00	2.0
	7.00	49.00	14.00	2.0
	8.00	64.00	16.00	2.0
	9.00	81.00	18.00	2.0
	10.00	100.00	20.00	2.0

SLROMB.

ROMBERG INTEGRATION

Integration of a user-defined function in non-tabular form i.e. evaluation of $\int_A^B f(x) dx$

where $f(x)$ is defined on the interval $[A,B]$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PROMB1, PROMB2 PROMB3 PROMB4	start and end of interval of integration PROMB2 > PROMB1 maximum number of lines of triangular scheme to be calculated tolerance	
OUTPUT		
I1 K! FUN.	value of the integral number of lines of triangular scheme calculated Return Status	Ø = correct calculation 1 = tolerance not reached after PROMB3 iterations 2 = PROMB3 not a posi- tive integer 3 = tolerance not posi- tive 4 = start of interval ≥ end

CALCULATION	W(PROMB3+1, PROMB3+1), S
CALLING SEQUENCE	GOSUB 45001
METHOD	See Appendix A
CALLED SUBROUTINES	

Note

The function to be integrated must be included in the calling program as FNF. For example $f(x) = e^x$ would be coded as

```
DEF FNF(X) = EXP(X)
```


NUMERICAL INTEGRATION AND DIFFERENTIATION

TEST PROGRAM FOR SLROMB.

TEST PROGRAM NAME: SLROMB	
Note: User-defined function $20 \text{ DEF FNF}(x) = \text{EXP}(x)$	
MERGE: SLROMB.	
INPUT	OUTPUT
ENTER START,END INTERVAL ? 0,1 ENTER MAXIMUM# OF ITERATIONS ? 2 ENTER TOLERANCE ? .000001	INTERVAL [0 1] MAXIMUM# OF ITERATIONS = 2 TOLERANCE = .000001 2 STEPS - NO SOLUTION YET LAST VALUE = 1.71886110305786
ENTER START,END INTERVAL ? 0, 1 ENTER MAXIMUM # OF ITERATIONS ? 15 ENTER TOLERANCE ? .000001	INTERVAL [0 1] MAXIMUM# OF ITERATIONS = 15 TOLERANCE = .000001 4 STEPS - VALUE OF INTEGRAL = 1.718288182210367
ENTER START, END INTERVAL ?	

GAUSS - LEGENDRE QUADRATURE

Integration of a user-defined function given in a non-tabular form i.e. evaluation of $\int_A^B f(x) dx$

where $f(x)$ is a function defined on the interval $[A,B]$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PGAUS1 PGAUS2 X(I),W(I)	order (N) of quadrature start and end of interval of integration $B > A$ zeros of the LEGENDRE, poly- nomials and corresponding weights $I = 1, \dots, 107$. $I = \sum_{i=1}^{N-1} i'$ to $\left(\sum_{i=1}^N i'\right) - 1$, where i' is the integer part of $(i+1)/2$ is data relevant to an order N quad- rature. Data is read from WEIGH1 (as shown in the test program)	$N = 2, 3 \dots 16, 24, 48$
OUTPUT		
I9 FUN.	value of Integral Return Status	$\emptyset$ = correct calculation 1 = invalid order of quadrature 2 = start of interval $\geq$ end

NUMERICAL INTEGRATION AND DIFFERENTIATION

CALLING SEQUENCE GOSUB 45201
METHOD See Appendix A
CALLED SUBROUTINES

Notes

1. The user-defined function $f(x)$ must be included in the calling program as FNF. For example, $f(x) = x^2$ would be coded as

DEF FNF(X) = X * X

2. The abscissae and weight factors for Gaussian integration are taken from: M. Abramowitz and I. Segun - Handbook of Mathematical functions (page 916 - 917)

TEST PROGRAM FOR SLGAUS.

TEST PROGRAM NAME: SLGAUS	
Note: In this program if the order of quadrature (PGAUS1) = 9D99 it is possible to change the start and end of the interval (PGAUS2). User-defined function: 2Ø DEF FNF(x) = SIN(x)	
MERGE: SLGAUS.	
INPUT	OUTPUT
ENTER START,END OF INTERVAL ? Ø, 3.14159265 ENTER ORDER OF QUADRATURE ? 3 ENTER ORDER OF QUADRATURE ? 5 ENTER ORDER OF QUADRATURE ? 9D99 ENTER START,END OF INTERVAL ?	ORDER3 QUADRATURE INTERVAL [Ø 3.14159267] VALUE OF INTEGRAL = 2ØØ138897538472 ORDER5 QUADRATURE INTERVAL [Ø 3.14159265] VALUE OF INTEGRAL = 2ØØØØØØØ18615244

GAUSS - LAGUERRE QUADRATURE

To evaluate $\int_0^{\infty} e^{-x} f(x) dx$ or $\int_0^{\infty} f(x) dx$

where $f(x)$ is a user-defined function in non-tabular form

INPUT		
VARIABLE	DESCRIPTION	VALUE
PLAGU1 PLAGU2 X(I), W(I,1), W(I,2)	order of quadrature (N) type of integrand zeros of the Laguerre polynomials and corresponding weights W and W^i $I = \sum_{i=1}^{N-1} i$ to $\left(\sum_{i=1}^N i \right) - 1$ is data relevant to an order N quadrature. Data is read from WEIGH2 (as shown in the test program)	N = 2,3,...10 only $1 = e^{-x} f(x)$ $2 = f(x)$ I = 1,...54 see Method
OUTPUT		
I9 FUN.	value of integral Return Status	0 = correct calculation 1 = invalid order of quadrature 2 = type of integrand other than 1 or 2

CALLING SEQUENCE GOSUB 45401
METHOD See Appendix A
CALLED SUBROUTINES

Notes

1. The user-defined function $f(x)$ must be included in the calling program as FNF. For example, $f(x) = e^x$ would be coded as

```
DEF FNF(X) = EXP(X)
```

2. The abscissae and weight factors for Laguerre Interpretation are taken from: M. Abramowitz and I. Segun - Handbook of Mathematical Functions (page 923)

NUMERICAL INTEGRATION AND DIFFERENTIATION

TEST PROGRAM FOR SLLAGU.

TEST PROGRAM NAME: SLLAGU	
Note: In this program if the order of quadrature (PLAGU1) = 9D99 it is possible to change the type of integrand. User-defined Function 20 DEF FNF(X) = EXP(X)	
MERGE: SLLAGU.	
INPUT	OUTPUT
ENTER TYPE OF INTEGRAND ? 1 ENTER ORDER OF QUADRATURE ? 3 ENTER ORDER OF QUADRATURE ? 4 ENTER ORDER OF QUADRATURE ? 9D99 ENTER TYPE OF INTEGRAND ? 2 ENTER ORDER OF QUADRATURE ? 5 ENTER ORDER OF QUADRATURE ? 10 ENTER ORDER OF QUADRATURE ? 11 ENTER ORDER OF QUADRATURE ?	ORDER3 QUADRATURE TYPE 1 VALUE OF INTEGRAL = 1.9999998738314 ORDER4 QUADRATURE TYPE 1 VALUE OF INTEGRAL = 2.000000006725262 ORDER5 QUADRATURE TYPE 2 VALUE OF INTEGRAL = 1409.3774 ORDER10 QUADRATURE TYPE 2 VALUE OF INTEGRAL = 14133.204 ORDER11 QUADRATURE TYPE 2 ORDER OF QUADRATURE 2, 3...10 ONLY

cc

c

c

c

cc

12. ORDINARY DIFFERENTIAL EQUATIONS

ABOUT THIS CHAPTER

This chapter contains a subroutine for the solution of Ordinary Differential Equations.

CONTENTS

SLEROM. Integration of Dif- 12-1
ferential Equations
(Euler-Romberg
Method)

**INTEGRATION OF DIFFERENTIAL EQUATIONS
(EULER - ROMBERG METHOD)**
SLEROM.

To integrate over an interval $[x_0, x_1]$ a system of first order ordinary differential equations with initial conditions

$$y'_i(x) = f_i[x, y_1(x), \dots, y_N(x)]$$

$$y_i(x_0) = y_0 \quad i = 1, \dots, N$$

or an Nth order ordinary differential equation with initial conditions

$$y^{(N)}(x) = g[x, y(x), y'(x), \dots, y^{(N-1)}(x)]$$

$$y(x_0) = y_0, \quad \frac{d^i y(x_0)}{dx^i} = y_i \quad i = 1, \dots, N-1$$

INPUT		
VARIABLE	DESCRIPTION	VALUE
PEROM1, PEROM2 PEROM3	start and end of interval of integration, PEROM1 < PEROM2 number of points at which the solution is to be calculated i.e. the step size $h = (PEROM2 - PEROM1)/PEROM3$ and the solution is given at $X_0+h, X_0+2h, \dots, PEROM2$	PEROM1 = X_0
PEROM4	order of system/equation (N)	
PEROM5	maximum number of step-size halvings	
PEROM6	tolerance, $PEROM6 \geq 10^{-9}$	
PEROM7	system/equation switch	1 = first order system 2 = single, higher order equation
Y(N)	The N initial conditions i.e. $Y_i(X_0), i = 1, \dots, N$ or $Y^{(i)}(X_0), i = 0, 1, \dots, N-1$ $Y()$ is destroyed during calculation	

OUTPUT		
S(PEROM3, N+1) K7 FUN.	solution at each of the PEROM3 points requested. S(I,J), J=1, ...,N; I = 1,...,PEROM3 is either $Y_j(XI)$ or $Y^{(j-i)}(XI)$ depending on PEROM7. S(I,N+1), I = 1,... PEROM3 equals zero if the required tolerance has been reached for the solution at XI, and equals unity if not. total number of system evaluations Return Status	Ø = correct calculation 1 = start of interval ≥ end 2 = order other than a positive integer 3 = maximum number of step-size halvings other than a positive integer 4 = number of points other than a positive integer 5 = tolerance less than 10^{-9} 6 = PEROM7 other than 1 or 2
CALCULATION VEROM3(N,22), VEROM2(N), H9, K2!, K4, K8, J1!, J4!, X, X2, PB1, PB2 PB3, PB4		
CALLING SEQUENCE GOSUB 456Ø1		
METHOD See Appendix A		
CALLED SUBROUTINES		

ORDINARY DIFFERENTIAL EQUATIONS

Note

The user must include the system of first order equations or the single, higher order equation in the calling program as a subroutine whose first line is the line 20. For example, the system:

$$y'_1 = y_1^2 y_2$$

$$y'_2 = \frac{-1}{y_1}$$

would be coded as:

$$20 \text{ F}(1) = Y(1) * Y(1) * Y(2)$$

$$30 \text{ F}(2) = -1/Y(1)$$

40 RETURN

The second order equation

$$Y'' = \frac{-Y'}{X} - Y$$

would be coded as:

$$20 \text{ F}(2) = -F(1)/X - Y(1)$$

30 RETURN

TEST PROGRAM FOR SLEROM.

TEST PROGRAM NAME: SLEROM	
Note: User-defined system $2\emptyset (F1) = Y(1)^2 * Y(2)$ $3\emptyset (F2) = -1/Y(1)$ $4\emptyset$ RETURN And for equation (second example) $2\emptyset F(2) = -F(1)/X - Y(1)$ $4\emptyset$ RETURN	
MERGE: SLEROM.	
INPUT	OUTPUT
ENTER1 (SYSTEM) or 2 (EQUATION) ? 1 ENTER ORDER ? 2 ENTER INITIAL, FINAL X ? $\emptyset$, 3 ENTER Y1 ($\emptyset$) ? 1 ENTER Y2 ($\emptyset$) ? 1 ENTER # OF POINTS ? $3\emptyset$ ENTER TOLERANCE .$\emptyset\emptyset\emptyset\emptyset\emptyset 1$ ENTER MAX # OF HALVINGS ? 5	ORDER 2 SYSTEM INTERVAL [$\emptyset$ 3] INITIAL CONDITION $x = \emptyset$ $Y1(x) = 1$ $Y2(x) = 1$ INTERMEDIATE POINTS $3\emptyset$ MAX # OF HALVINGS 5 TOLERANCE .$\emptyset\emptyset\emptyset\emptyset\emptyset 1$ SOLUTION (see next page)

ORDINARY DIFFERENTIAL EQUATIONS

INPUT	OUTPUT
	<pre> X = .1 Y 1 CK0 = 1.105170912089 Y 2 CK0 = .90483741782036 X = .2 Y 1 CK0 = 1.22140274711143 Y 2 CK0 = .818730752268603 X = .3 Y 1 CK0 = 1.34885879562493 Y 2 CK0 = .748818219357045 X = .4 Y 1 CK0 = 1.49182468849357 Y 2 CK0 = .670320044069804 X = .5 Y 1 CK0 = 1.64872125538999 Y 2 CK0 = .585530655722211 X = .6 Y 1 CK0 = 1.822118795538969 Y 2 CK0 = .548811633192593 X = .7 Y 1 CK0 = 2.01375270820213 Y 2 CK0 = .495585300871541 X = .8 Y 1 CK0 = 2.22554092804031 Y 2 CK0 = .44832886111134 X = .9 Y 1 CK0 = 2.4566031039484 Y 2 CK0 = .40656935650931 X = 1.0 Y 1 CK0 = 2.7182818211462 Y 2 CK0 = .367879377823783 X = 1.1 Y 1 CK0 = 3.004659952215 Y 2 CK0 = .33287108011523 X = 1.2 Y 1 CK0 = 3.3201166292672 Y 2 CK0 = .301194207861791 X = 1.3 Y 1 CK0 = 3.6692665910452 Y 2 CK0 = .272531788535788 X = 1.4 Y 1 CK0 = 4.05519986614765 Y 2 CK0 = .246596350857881 X = 1.5 Y 1 CK0 = 4.48168892605484 Y 2 CK0 = .223138154396473 X = 1.6 Y 1 CK0 = 4.95303223454992 Y 2 CK0 = .201896511520333 X = 1.7 Y 1 CK0 = 5.47394720441388 Y 2 CK0 = .182683516920369 X = 1.8 Y 1 CK0 = 6.04964719397352 Y 2 CK0 = .165296880372421 X = 1.9 Y 1 CK0 = 6.68589406866321 Y 2 CK0 = .149568610583411 X = 2.0 Y 1 CK0 = 7.38905554716329 Y 2 CK0 = .135335573682362 X = 2.1 Y 1 CK0 = 8.16616916502761 Y 2 CK0 = .12245641762619 X = 2.2 Y 1 CK0 = 9.02501249219802 Y 2 CK0 = .110803146567096 X = 2.3 Y 1 CK0 = 9.97416107646877 Y 2 CK0 = .100258630609724 X = 2.4 Y 1 CK0 = 11.0231745845235 Y 2 CK0 = 9.07179287446330-02 X = 2.5 Y 1 CK0 = 12.1824916403933 Y 2 CK0 = 8.20849825603290-02 X = 2.6 Y 1 CK0 = 13.4637749084043 Y 2 CK0 = 7.427355051863890-02 X = 2.7 Y 1 CK0 = 14.87972749742 Y 2 CK0 = .067205493224391 X = 2.8 Y 1 CK0 = 16.4445410182883 Y 2 CK0 = 5.081004110916820-02 X = 2.9 Y 1 CK0 = 18.1741377143762 Y 2 CK0 = 5.502319632213150-02 X = 3.0 Y 1 CK0 = 20.0855265659384 Y 2 CK0 = .049787042160655 </pre>
ENTER INITIAL, FINAL X ? .1, 3,1 ENTER Y [0] [.1] ? .99750156 ENTER Y [1] [.1] ? -4.9937526d -2 ENTER # OF POINTS ? 30 ENTER TOLERANCE ? .000001 ENTER MAX# OF HALVINGS ? 5	500 DERIVATIVE EVALUATIONS INTERVAL [.1, 3, 1] INITIAL CONDITION X = .1 Y [-1] [X] = .99750156 Y [-1] [X] = -.049937526 INTERMEDIATE POINTS 30 MAX# OF HALVINGS 5 TOLERANCE .000001

INPUT	OUTPUT
	SOLUTION <pre> X = 2 YC 0 0 CK0 = 1880024970383424 YC 1 0 CK0 -- 099500083528-621 X = 3 YC 0 0 CK0 = 977626244512114 YC 1 0 CK0 -- 148318818907361 X = 4 YC 0 0 CK0 = 90388224364774 YC 1 0 CK0 -- 19602636140588 X = 5 YC 0 0 CK0 = 938469884634624 YC 1 0 CK0 -- 242158440888853 X = 5.5 YC 0 0 CK0 = 12004860621348 YC 1 0 CK0 -- 286700930541378 X = 6 YC 0 0 CK0 = 31200885438337 YC 1 0 CK0 -- 32899574503003 X = 6.5 YC 0 0 CK0 = 346267349130906 YC 1 0 CK0 -- 368842051627558 X = 7 YC 0 0 CK0 = 387523783814717 YC 1 0 CK0 -- 4055495542647 X = 7.5 YC 0 0 CK0 = 465197681278184 YC 1 0 CK0 -- 4400500869145 X = 8 YC 0 0 CK0 = 518622011868606 YC 1 0 CK0 -- 470980248917489 X = 8.5 YC 0 0 CK0 = 571132736112044 YC 1 0 CK0 -- 498288053014111 X = 9 YC 0 0 CK0 = 620065876506768 YC 1 0 CK0 -- 522023247894501 X = 9.5 YC 0 0 CK0 = 668853108123176 YC 1 0 CK0 -- 541847737250073 X = 10 YC 0 0 CK0 = 716217637017431 YC 1 0 CK0 -- 557937657380021 X = 10.5 YC 0 0 CK0 = 765402150211827 YC 1 0 CK0 -- 569035963114021 X = 11 YC 0 0 CK0 = 8137884839102768 YC 1 0 CK0 -- 577755261561103 X = 11.5 YC 0 0 CK0 = 862666387615506 YC 1 0 CK0 -- 581516983235016 X = 12 YC 0 0 CK0 = 911818532738128 YC 1 0 CK0 -- 581157105257148 X = 12.5 YC 0 0 CK0 = 960990749215337 YC 1 0 CK0 -- 576724840875648 X = 13 YC 0 0 CK0 = 1066806947164306 YC 1 0 CK0 -- 568292167381224 X = 13.5 YC 0 0 CK0 = 116862230745335 YC 1 0 CK0 -- 555963078325761 X = 14 YC 0 0 CK0 = 126553974563800660-02 YC 1 0 CK0 -- 539872586705503 X = 14.5 YC 0 0 CK0 = 13507642348773830-03 YC 1 0 CK0 -- 520185286921313 X = 15 YC 0 0 CK0 = 14383381888647080-02 YC 1 0 CK0 -- 497034115170438 X = 15.5 YC 0 0 CK0 = 152680438785970650-02 YC 1 0 CK0 -- 470818272771808 X = 16 YC 0 0 CK0 = 162443413797147 YC 1 0 CK0 -- 441601378725753 X = 16.5 YC 0 0 CK0 = 172503607673861 YC 1 0 CK0 -- 409709239815982 X = 17 YC 0 0 CK0 = 1824311588132895 YC 1 0 CK0 -- 375427468308357 X = 17.5 YC 0 0 CK0 = 19260051995578783 YC 1 0 CK0 -- 339858938881856 X = 18 YC 0 0 CK0 = 202864386071678 YC 1 0 CK0 -- 300921187772824 900 DERIVATIVE EVALUATIONS </pre>
ENTER INITIAL, FINAL X ?	

A. CALCULATION METHODS

ABOUT THIS CHAPTER

This chapter contains the calculation methods used by the OLINUM subroutines.

CONTENTS

<u>COMBINATORIAL ANALYSIS</u>	A-1	<u>ELEMENTARY FUNCTIONS</u> <u>(COMPLEX)</u>	A-7
SLCONN.	A-1	SLCSIN.	A-7
SLPRIM.	A-2	SLCOS.	A-7
SLEUCL.	A-2	SLCTAN.	A-8
SLRFCO.	A-3	SLCASN.	A-8
SLSUCO.	A-3	SLCACS.	A-8
SLCFCO.	A-4	SLCATN.	A-8
SLFACT.	A-4	SLCLN.	A-9
SLBINO.	A-5	SLCEXP.	A-9
SLMULT.	A-5	SLCRZ.	A-9
SLDUPL.	A-6	SLCZMZ.	A-9
<u>ELEMENTARY FUNCTIONS</u>	A-6	SLCZDZ.	A-10
SLRPCC.	A-6	SLCSQR.	A-10
SLPRCC.	A-6	SLCZN.	A-10

SLCZA.	A-11	SLEIF.	A-15
<u>POLYNOMIALS</u>	<u>A-11</u>	SLINF.	A-16
SLPLRC.	A-11	<u>SOLUTION OF EQUATIONS</u>	A-16
SLPLRR.	A-11	SLBAIR.	A-16
SLPRRR.	A-11	SLRBIS	A-16
SLPLYM.	A-12	SLNLIN.	A-17
<u>HIGHER MATHEMATICAL FUNCTIONS</u>	<u>A-12</u>	<u>LINEAR ALGEBRA</u>	A-18
SLKMF.	A-12	SLCRO.	A-18
SLEMF.	A-12	SLCHO.	A-18
SLLAGG.	A-12	SLJAC.	A-18
SLHNF.	A-13	SLHES.	A-19
SLHEN.	A-13	SLHEV.	A-19
SLFOUR.	A-13	SLCHA.	A-20
SLGAMA.	A-13	SLJCB.	A-20
SLERF.	A-13	<u>CURVE FITTING AND INTERPOLATION</u>	A-20
SLBJN.	A-14	SLLLSQ.	A-20
SLIIØX.	A-14	SLNLLS.	A-22
SLSF.	A-14	SLFSYT.	A-25
SLCF.	A-14	SLFTRP.	A-26
SLCHYF.	A-15	SLFOUI.	A-28
SLGHYP.	A-15	SLPLYF.	A-29
SLSIF.	A-15	SLCSPC.	A-31
SLCINF.	A-15	SLCSPI.	A-32

SLPADE. A-32

NUMERICAL INTEGRATION A-34
AND DIFFERENTIATION

SLITEQ. A-34

SLDTEQ. A-35

SLDIFF. A-36

SLROMB. A-36

SLGAUS. A-37

SLLAGU. A-38

ORDINARY DIFFERENTIAL A-38
EQUATIONS

SLEROM. A-38

COMBINATORIAL ANALYSIS

SLCONN.

Since the integer and fractional parts are converted separately they are supplied to the function as separate parameters so as to allow a maximum of 13 significant digits for each. The digits of the converted number will each be represented by two decimal digits for bases greater than 10, three for bases greater than 100, and so on. Input of numbers to bases greater than 10 must also follow this convention.

Consider conversion from base 10 to base b, and let N be the decimal number and B the corresponding number to base b.

Let

$$I_N, I_B, D_N, D_B$$

be the whole and decimal fraction part of N and B respectively.

Step 1: Calculate the modulus, m, of base b, i.e. the smallest power of 10 greater than or equal to 10. For example, the moduli of 2, 8 and 16 are 10, 10 and 100 respectively.

Step 2: The following integral divisions are carried out:

$$I_N/b = Q_0, \text{ remainder } R_0$$

$$Q_0/b = Q_1, \text{ remainder } R_1$$

$$\begin{array}{cccc} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{array}$$

Division terminates when $Q = 0$.

Then

$$I_B = R_0 + mR_1 + m^2 R_2 + \dots m^n R_n$$

Step 3:

$$D_N \times b = i_1 \text{ (whole part) and } d_1 \text{ (decimal part); } \frac{i_1}{m}$$

$$d_1 \times b = i_2 \text{ (whole part) and } d_2 \text{ (decimal part); } \frac{i_2}{m^2}$$

$$\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{array}$$

The operation is completed when $d_n = \emptyset$ or when 13 significant digits have been generated for D_B , where

$$D_B = \frac{1}{m} + \frac{i_2}{m^2} + \frac{i_3}{m^3} + \dots + \frac{i_n}{m^n}$$

To convert from base 10, the above procedure is used with b and m interchanged. The conversion of the fractional parts is completed when all the significant lines have been evaluated. To convert from base b1 to base b2 (b1, b2, #10).

When converting to a base 6 through 9 the last digit may appear to be incorrect i.e. equal to the base, due to rounding up. This situation can also occur when converting to a base >10.

SLPRIM.

The prime factors of any positive integer may be determined by dividing by 2, 3 and then all odd numbers between 5 and the square root of the number, or the last quotient found. The number is prime if no remainder less divisors are found. If a prime number occurs more than once as a factor, the appropriate exponent is returned.

SLEUCL.

Euclid's algorithm is used to evaluate the G.C.D of N1 and N2. The L.C.M is then given by $N1.N2/G.C.D$. Euclid's algorithm is the following recurrence:

$$\frac{P_k}{Q_k} = I_k + \frac{R_k}{Q_k}, \quad I_k \quad \text{is the integral part of the quotient}$$

$$R_k \quad \text{is the integral remainder}$$

$$P_{q+1} = R_k$$

$$Q_{k+1} = P_k$$

$$P_0 = N1$$

$$Q_0 = N2$$

When $R_n = \emptyset$, the G.C.D. is given by q_n .

SLRFCO.

The rational fraction $\frac{A}{B}$ may be represented by the continued fraction

$$\frac{A}{B} = b_0 + \frac{1}{b_{1+}} \frac{1}{b_{2+}} \cdot \cdot \cdot \frac{1}{b_n}$$

where

$$b_i = \left(\frac{r_i}{r_{i+1}} \right)_I, \text{ and } \left(\frac{r_i}{r_{i+1}} \right)_I \text{ indicates the integer part of } \frac{r_i}{r_{i+1}} \text{ and } r_{i+1} \text{ the remainder part of } \frac{r_i}{r_{i+1}}$$

This representation terminates after a finite number of terms.

SLSUCO.

The quadratic surd, $\sqrt{D}$, may be represented by the continued fraction

$$\sqrt{D} = b_0 + \frac{1}{b_{1+}} \frac{1}{b_{2+}} \cdot \cdot \cdot \frac{1}{b_{n+}} \cdot \cdot \cdot$$

where

$$b_i = \left(\frac{b_0 + p_i}{q_i} \right)_I, ()_I \text{ denotes the integer part of the expression in brackets.}$$

$$P_i = b_{i-1} q_i - p_{i-1}$$

$$q_i = (D - p_i^2)/q_{i-1}$$

$$b_0 = (\sqrt{D})_I$$

$$p_0 = 0$$

$$q_0 = 1$$

This representation is non-terminating, but repeats after a finite number of terms.

SLCFCO.

Given the continued fraction

$$b_0 + \frac{a_1}{b_{1+}} \frac{a_2}{b_{2+}} \frac{a_3}{b_{3+}} \dots$$

the first convergent is given by

$$b_0 + \frac{a_1}{b_1},$$

the second by

$$b_0 + \frac{a_1}{b_{1+}} \frac{a_2}{b_2},$$

and so on.

The successive convergents are calculated using the following recurrence:

$$p_i = b_i p_{i-1} + a_i p_{i-2}$$

$$q_i = b_i q_{i-1} + a_i q_{i-2}$$

$$p_0 = b_0, \quad p_{-1} = 1$$

$$q_0 = 1, \quad q_{-1} = 0 \quad \text{The } i^{\text{th}} \text{ convergent is then } \frac{p_i}{q_i}$$

SLFACT.

N! is evaluated in accordance with N(N-1) (N-2)...2.1.

N! may only be evaluated for $N \leq 170$ since larger values would cause numeric overflow, i.e. N! 1.797693134862D 308

Stirling's Formula for LnN! is the following:

$$\ln N! = \ln 2 + \ln(N+.5) - N + \ln k,$$

where

$$k = 1 - \frac{1}{12 N} - \frac{1}{288 N^2} - \frac{139}{51840 N^3} - \frac{571}{2488320 N^4}$$

$$= 1 + \frac{1}{12 N} \left(1 + \frac{1}{12 N} \left(\frac{1}{2} + \frac{1}{12 N} \left(\frac{-139}{30} + \frac{1}{12 N} \left(\frac{-571}{120} \right) \right) \right) \right)$$

This approximation of $\ln N!$ is useful for large values of N .

SLBIN0.

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

is evaluated using the recurrence:

$$\binom{n}{r} = \binom{n}{r-1} \left(\frac{n-r+1}{r} \right)$$

$$\binom{n}{0} = 1$$

Note: $\binom{n}{r} = \binom{n}{n-r}$

SLMULT.

$$(n; n_1, n_2, \dots, n_s) = \frac{n!}{n_1! n_2! \dots n_s!} \frac{(n_1 + 1)(n_1 + 2) \dots n}{n_2! n_3! \dots n_s!}$$

where

$$n = \sum_{i=1}^s n_i$$

is evaluated using the following recurrence:

$$p_0 = 1, q_1 = n_2$$

for $i = 1, 2, \dots, s$

$$p_j = p_{j-1} (q_i + j)/j \quad j = 1, 2, \dots, n_i$$

$$p_0 = p_{n_i}$$

$$q_{i+1} = q_i + n_{i+1}$$

$$(n; n_1, n_2, \dots, n_s) = p_{n_s}$$

SLDUPL.

$$p_k = 1 - \frac{p_{n-k-1}}{n^k}$$

$$= 1 - \frac{(n-1)(n-2) \dots (n-k-1)}{n^{k-1}}$$

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

is evaluated using the recurrence:

$$b_0 = 1$$

$$b_i = \frac{n-i}{n} \cdot b_{i-1} \quad i = 1, 2, \dots, k-1$$

$$p_k = 1 - b_{k-1}$$

ELEMENTARY FUNCTIONS

SLRPCC.

Given $z = x + iy$, then:

$$- \rho = \sqrt{x^2 + y^2} \text{ (modulus)}$$

$$- \vartheta = \operatorname{atn} \left(\frac{y}{x} \right) \text{ (phase)}$$

SLPRCC.

Given $z = \rho e^{i\vartheta}$, then:

$$x = \rho \cos \vartheta \quad \text{(real part)}$$

$$y = \rho \sin \vartheta \quad \text{(imaginary part)}$$

ELEMENTARY FUNCTIONS (COMPLEX)

SLCSIN.

- rectangular coordinates:

$$\begin{aligned}\sin z &= \sin(x - i y) \\ &= \sin x \cosh y + i \sinh y \cos x \\ &= \sin x \left(\frac{e^y + e^{-y}}{2} \right) + i \cos x \left(\frac{e^y - e^{-y}}{2} \right)\end{aligned}$$

- polar coordinates:

$$\begin{aligned}z &= \rho e^{i\vartheta} \\ &= \rho \cos \vartheta + i \rho \sin \vartheta \\ &= x + i y\end{aligned}$$

$$|\sin z| = \sqrt{\frac{1}{2}(\cosh 2y - \cos 2x)}$$

$$\text{ph}(\sin z) = \arctan \left(\frac{\cos x}{\sin x} \cdot \frac{\sinh y}{\cosh y} \right)$$

SLCOS.

- rectangular coordinates:

$$\begin{aligned}\cos z &= \cos(x - i y) \\ &= \cos x \cosh y - i \sin x \sinh y \\ &= \cos x \left(\frac{e^y + e^{-y}}{2} \right) - i \sin x \left(\frac{e^y - e^{-y}}{2} \right)\end{aligned}$$

- polar coordinates:

$$\begin{aligned}z &= \rho e^{i\vartheta} \\ &= \rho \cos \vartheta + i \rho \sin \vartheta \\ &= x + i y\end{aligned}$$

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

$$\text{ph}(\cos z) = \arctan \left(\frac{\sin x}{\cos x} \cdot \frac{\sinh y}{\cosh y} \right)$$

SLCTAN.

- rectangular coordinates

$$\tan(z) = \tan(x + i y)$$

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

- polar coordinates

$$z = \rho e^{i\theta}$$

$$= \rho \cos \theta + i \rho \sin \theta$$

$$= x + i y$$

$$\tan z = \frac{\cosh 2y - \cos 2x}{\cosh 2y + \cos 2x}$$

$$\text{ph}(\tan z) = \arctan \left(\frac{\sinh 2y}{\sin 2x} \right)$$

SLCASN.

$$z = x + i y$$

$$\arcsin z = \arcsin \beta + i \ln(\alpha + (\alpha^2 - 1)^{1/2})$$

where

$$\alpha = \frac{1}{2}((x+1)^2 + y^2)^{1/2} + \frac{1}{2}((x-1)^2 + y^2)^{1/2}$$

and

$$\beta = \frac{1}{2}((x+1)^2 + y^2)^{1/2} - \frac{1}{2}((x-1)^2 + y^2)^{1/2}$$

SLCACS.

$$z = x + i y$$

$$\arcsin z = \arccos \beta - i \ln(\alpha + (\alpha^2 - 1)^{1/2})$$

where

$$\alpha = \frac{1}{2}((x+1)^2 + y^2)^{1/2} + \frac{1}{2}((x-1)^2 + y^2)^{1/2}$$

and

$$\beta = \frac{1}{2}((x+1)^2 + y^2)^{1/2} - \frac{1}{2}((x-1)^2 + y^2)^{1/2}$$

SLCATN.

$$z = x + i y$$

$$\arctan z = \frac{1}{2} \arctan \left(\frac{2x}{1-x^2-y^2} \right) + \frac{i}{4} \ln \left(\frac{x^2 + (y+1)^2}{x^2 + (y-1)^2} \right), z^2 \neq -1$$

SLCLN.

$$\log(z) = \log(x + i y) = \log(\sqrt{x^2 + y^2}) - i \arctan\left(\frac{y}{x}\right), |z| > 0,$$

$$z = \varrho e^{i\vartheta}$$

$$|\log z| = \sqrt{(\log \varrho)^2 + \vartheta^2}$$

$$\text{ph}(\log z) = \arctan\left(\frac{\vartheta}{\log \varrho}\right)$$

SLCEXP.

$$z = x + i y$$

$$e^z = e^x e^{iy}$$

$$= e^x \cos y + i e^x \sin y$$

$$z = \varrho e^{i\vartheta}$$

$$e^z = e^{\varrho e^{i\vartheta}}$$

$$|e^z| = e^{\varrho \cos \vartheta}$$

$$\text{ph}(e^z) = \varrho \sin \vartheta$$

SLCRZ.

$$\begin{aligned} \frac{1}{z} &= \frac{1}{x + i y} \\ &= \frac{x - i y}{x^2 + y^2} \\ &= \frac{x}{x^2 + y^2} - i \frac{y}{x^2 + y^2}, |z| > 0 \end{aligned}$$

$$\begin{aligned} \left| \frac{1}{z} \right| &= \left| \frac{1}{\varrho e^{i\vartheta}} \right| \\ &= \frac{1}{\varrho} \end{aligned}$$

$$\begin{aligned} \text{ph}\left(\frac{1}{z}\right) &= \text{ph}\left(\frac{1}{\varrho e^{i\vartheta}}\right), |z| > 0 \\ &= -\vartheta \end{aligned}$$

SLCZMZ.

$$\begin{aligned} z_1 z_2 &= (x_1 + i y_1)(x_2 + i y_2) = x_1 x_2 - y_1 y_2 + i(y_1 x_2 + y_2 x_1) \\ &= x_1 x_2 - y_1 y_2 + i(y_1 x_2 + y_2 x_1) \end{aligned}$$

$$|z_1 z_2| = |\varrho_1 e^{i\vartheta_1} \cdot \varrho_2 e^{i\vartheta_2}|$$

$$= \varrho_1 \varrho_2$$

$$\text{ph}(z_1 z_2) = \vartheta_1 + \vartheta_2$$

SLCZDZ.

Let

$$z_1 = a + i b = \varrho_1 e^{i\vartheta_1}$$

$$z_2 = c + i d = \varrho_2 e^{i\vartheta_2}$$

$$\frac{z_1}{z_2} = \frac{a + i b}{c + i d} \cdot \frac{c - i d}{c - i d} = \frac{(a c + b d) + i(b c - a d)}{c^2 + d^2}$$

$$\left| \frac{z_1}{z_2} \right| = \frac{\varrho_1}{\varrho_2}$$

$$\text{ph}\left(\frac{z_1}{z_2}\right) = \vartheta_1 - \vartheta_2.$$

SLCSQR.

Let

$$z = x + i y$$

$$= \varrho e^{i\vartheta}$$

$$\sqrt{z} = \sqrt{\frac{\sqrt{x^2 + y^2} + x}{2}} + i \frac{y}{|y|} \sqrt{\frac{\sqrt{x^2 + y^2} - x}{2}}$$

$$|\sqrt{z}| = \sqrt{\varrho}$$

$$\text{ph}(\sqrt{z}) = \frac{\vartheta}{2}$$

SLCZN.

Let

$$z = x + i y = \varrho e^{i\vartheta}$$

$$z^n = z z^{n-1};$$

CALCULATION METHODS

$$z^n = x_n + i y_n$$

$$x_n = x x_{n-1} - y y_{n-1} \quad x_0 = 1$$

$$y_n = x y_{n-1} + y x_{n-1} \quad y_0 = 0$$

$$|z^n| = \rho^n$$

$$\text{ph}(z^n) = n\vartheta$$

SLCZA.

Let

$$z = x + i y$$

$$= \rho e^{i\vartheta}$$

$$z^a = |z|^a e^{i a \text{ph}(z)}$$

$$= |z|^a \cos(a \text{ph}(z)) + i |z|^a \sin(a \text{ph}(z)).$$

$$|z^a| = \rho^a$$

$$\text{ph}(z^a) = a \vartheta$$

POLYNOMIALS

SLPLRC.

Horner Scheme

SLPLRR.

Horner Scheme

SLPRRR.

The coefficients are computed by multiplying the roots and summing up the terms of the same degree.

SLPLYM.

$$\sum_{i=0}^N a_i x^i * \sum_{i=0}^M b_i x^i = \sum_{i=0}^{M+N} c_i x^i$$

where

$$c_i = \sum a_h b_k \quad \text{with} \quad h + k = i$$

HIGHER MATHEMATICAL FUNCTIONS

SLKMF.

The value is computed by means of the representation

$$K(x) = \frac{1}{2} \pi \left[1 + \left(\frac{1}{2} \right)^2 x + \left(\frac{1 \cdot 3}{2 \cdot 4} \right)^2 x^2 + \left(\frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \right)^2 x^3 + \dots \right]$$

This series is valid for $|x| < 1$, but speed of convergence decreases as $|x| \rightarrow 1$.

SLEMF.

The value is computed by means of the representation

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

This series is valid for $|x| < 1$, but speed of convergence decreases as $|x| \rightarrow 1$.

SLLAGG.

The value $L_n^{(a)}(x)$ is computed by means of the recurrence relation

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

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

SLHNF.

The value is computed by means of the recurrence relation

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

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

SLHEN.

The value is computed by means of the recurrence relation

$$H e_0(x) = 1, \quad H e_1(x) = x$$

$$H e_n(x) = x H e_{n-1}(x) - (n-1) H e_{n-2}(x)$$

SLFOUR.

The following expression is used for calculation

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

SLGAMA.

For x in the range of $0 \leq x \leq 1$ an approximating polynomial is used to calculate $\gamma(1+x)$. The associated error is less than 3×10^{-7} in absolute value. Outside this interval the following recurrence formulae are used.

$$\Gamma(x-1) = x \Gamma(x)$$

$$\Gamma(n-x) = (n-1-x)(n-2-x) \dots (1-x) \Gamma(1-x)$$

$$\Gamma(x-n) = \frac{\Gamma(1-x)}{x(x-1) \dots (x-n)}$$

SLERF.

An approximation series is used

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

As

$$|x| \rightarrow \infty, \quad |\operatorname{erf}| \rightarrow 1.$$

A value of 1 is returned for $|\operatorname{erf}(x)|$ for $|x| \geq 5$.

SLBJN.

The value is computed by means of the series representation:

$$J_n(x) = \left(\frac{1}{2}x\right)^n \sum_{k=0}^{\infty} \frac{\left(-\frac{1}{4}x^2\right)^k}{k! \Gamma(n+k+1)}, \quad x \text{ not } \gg n.$$

$$J_{-n}(x) = (-1)^n J_n(x)$$

SLIIOX.

The following approximation is used:

$$f(x) = x \left(1 + \sum_{n=1}^{\infty} \frac{\left(\frac{x^2}{4}\right)^n}{(n!)^2 (2n+1)} \right)$$

SLSF.

An approximation series is used:

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

SLCF.

A series approximation is used:

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

SLCHYF.

The value is computed by means of the representation:

$$M(a, b, x) = \sum_{n=0}^{\infty} \frac{(a)_n}{(b)_n} \cdot \frac{x^n}{n!} \quad b \neq 0, -1, -2, \dots,$$

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

SLGHYP.

The value is computed by means of the representation:

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

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

For a or b a negative integer or zero, this series reduces to a polynomial in x. The series is not defined for c a negative integer or zero if a or b is not a negative integer or zero greater than c. For a or b negative integer or zero, |x| must be <1 or ≤ 1 if c - a - b > 0

SLSIF.

Si(x) is evaluated using the series expansion:

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

SLCINF.

Cin(x) is evaluated using the series expansion:

$$\text{Cin}(x) = - \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{2n(2n)!}$$

SLEIF.

$$E_1(x) = \gamma + \ln x + \sum_{n=1}^{\infty} \frac{x^n}{n n!}$$

where γ is the Euler constant 0.5772156649...

SLEINF.

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

SOLUTION OF EQUATIONS

SLBAIR.

A quadratic factor of the polynomial is determined, then the roots of this factor are computed. Let $f(x)$ be the polynomial, $g_k(x)$ and approximation to a quadratic factor of $f(x)$:

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

$$g_K(x) = x^2 - p_K x - q_K$$

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

$$p_{K+1} = p_K + \Delta p \quad q_{K+1} = q_K + \Delta q$$

with

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

$$\left. \begin{aligned} b_K &= b_{K+1} p_K + b_{K+2} q_K + a_K \\ c_K &= c_{K+1} p_K + c_{K+2} q_K + b_K \end{aligned} \right\} \quad 0 \leq K \leq n$$

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

The algorithm stops when

$$1. \quad x^2 - p_{K+1} x - q_{K+1}$$

E : user-defined tolerance

$$|\Delta p| \quad \text{and} \quad |\Delta q| < E \quad \text{is said to be quadratic factor, or}$$

2. more than Z iterations are performed without 1 being satisfied

Z : user-defined maximum number of iterations.

SLRBIS.

The bisection method is also called the interval halving method. Let $[a, b]$ be an interval containing a root, $c = \frac{a+b}{2}$ is the midpoint of the interval.

CALCULATION METHODS

A test is made as to whether $[a,c]$ or $[c,b]$ contains a root. There is a root on the interval $[a,b]$ if $f(a).f(b) \leq 0$. The process is repeated on the appropriate half interval. A root has been found if:

$f(d) = 0$ for some end-point d of a subinterval

or

$|b-a| < E$ for some tolerance E .

SLNLIN.

Let $F_1, \dots, F_n$ be n not-necessarily-linear functions in the n variables $x_1, \dots, x_n$. The method used is the generalization to n -dimensions of the familiar Newton-Raphson method.

Let $x_j^{(i)}$ be the i th approximation to the j th component of a root

$$x_j^{(i+1)} = x_j^{(i)} + \Delta x_j \quad \text{for } j = 1, 2, \dots, n$$

where the Δx_i are the solution of the linear system:

$$-F_k(x_1^{(i)}, \dots, x_n^{(i)}) = \sum_{j=1}^n \frac{\partial F_k}{\partial x_j} \cdot \Delta x_j; \quad K = 1, \dots, n$$

$$k = 1, \dots, n$$

and $\frac{\partial F_k}{\partial x_j}$ is computed by considering F as a function of x only and using a 5 point central difference, i.e.:

$$\frac{\partial F_k}{\partial x_j} = \{F_k(x_1, \dots, x_j + 2h, \dots, x_n) - 8(F_k(x_1, \dots, x_j + h, \dots, x_n) + F_k(x_1, \dots, x_j - h, \dots, x_n)) + F_k(x_1, \dots, x_j - 2h, \dots, x_n)\}$$

$$\text{where } h = .01$$

A root has been found if for some i

$$\sqrt{\sum_j (\Delta x_j^{(i)})^2} < E \quad \text{for some tolerance } E.$$

If $\sqrt{\sum_j (\Delta x_j^{(i)})^2} > 1/E^2$ the algorithm diverges. Otherwise, the routine halts after a specified number of iterations.

LINEAR ALGEBRA

SLCRO.

The linear system $Ax = b$ is solved in two steps. The matrix A is first decomposed into a product of an upper and a lower triangular matrix, U and L , by Crout's algorithm with row interchanging, such that $A = LU$.

In a second step, the triangular system $Ly = b$ and $Ux = y$ are solved for the required vector x .

The inverse matrix A^{-1} is evaluated by solving LUx for the N columns of the identity matrix as right-hand sides. The N solution vectors are the columns of A^{-1} . The determinant of A is given (but for sign) by the product of the diagonal elements of L .

SLCHO.

The linear system $Ax = b$ is solved in two steps. The matrix A is first decomposed into a product of an upper and a lower triangular matrix, R and R' , by Cholesky's algorithm such that $A = R'R$.

In a second step, the triangular systems $R'y = b$ and $Rx = y$ are solved for the required vector x .

The inverse matrix A^{-1} is evaluated by solving $R'R x$ for the N columns of the identity matrix as right-hand sides. The N solution vectors are the columns of A^{-1} . The determinant of A is given (but for sign) by the square of the product of the diagonal element of R .

SLJAC.

The matrix equation $Ax = \lambda x$ is solved by means of the Jacobi method. Through a series of orthogonal similarity transformations the original matrix A is transformed into a matrix S in diagonal form. If the product of these orthogonal transformation matrices is called P , S is given by

$$S = P^T A P$$

Since the eigenvalues are preserved through an orthogonal transformation, the elements of the final diagonal matrix S must be the eigenvalues of the original matrix.

CALCULATION METHODS

The matrix P has the eigenvectors as its columns. Since the off-diagonal elements of S will not in practice be zero, the value of the greatest element is related to a user defined tolerance.

SLHES.

The calculation is accomplished in two steps. First the matrix A is reduced to upper Hessenberg form H by means of orthogonal similarity transformations.

$$H = P^T A P$$

In a second step, H is reduced to an upper triangular matrix U using the QR-algorithm. Let Q be the product of the transformation matrices, then

$$U = Q^T H Q$$

where the diagonal elements of U are the eigenvalues of A .

For calculation of eigenvectors, see SLHEV.

SLHEV.

Having calculated the eigenvalues of A according to the method described in SLHES., the problem is to find the matrix X which has the eigenvectors of A as its columns. The matrix X must satisfy the relation

$$X^T A X = D \quad (1)$$

where D is the diagonal matrix of the eigenvalues λ_i of U (and of A). the eigenvectors of U are the columns y of Y where Y satisfies the relation

$$Y^T U Y = D \quad (2)$$

The y_i are obtained by solving the linear homogeneous system

$$(U - \lambda_i I) y_i = 0$$

which is simplified due to the block triangular shape of U . Substituting $W^T A W$ for U in (2) gives

$$Y^T W^T A W Y = D$$

which, comparing with (1), gives

$$X = WY$$

SLCHA.

The coefficients of the characteristic polynomial

$$p(\lambda) = \det(A - \lambda I)$$

are computed by Leverrier's Method

For

$$i = 1, \dots, N$$

$$B_i = A(B_{i-1} - p_{i-1} I)$$

$$p_i = \text{tr}(B_i)/i$$

where

$$B_0 = 0, p_0 = 1$$

and

$$\text{tr}(B) = \sum_{j=1}^N b_{jj}$$

(N is the order of the matrix)

SLJCB.

The program used the five point differencing scheme to evaluate the $n \times n$ matrix $(\delta F_i / \delta x_j)$, where F_i is a user supplied function and

$$\frac{\delta F_i}{\delta x_j} = \frac{\delta F_i(x_1, x_2, \dots, x_j, \dots, x_n)}{\delta x_j}$$

CURVE FITTING AND INTERPLOTATION

SLLSQ.

Let (x_i, y_i) $i = 1, 2, \dots, N$ be 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

$$g_K(x) = \sum_{j=1}^K a_j f_j(x) \quad K \leq 6$$

is the approximating function of $f(x)$ and $f_j(x)$ 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 derivatives of E with respect to parameters a_j . In this way one obtains a linear system of K equations in the K unknowns $a_1, a_2, \dots, a_K$

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

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

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

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

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

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

By comparing the values of σ at each step, it is possible to evaluate the effect of the addition of each new basis function. The coefficients a_i are computed to the index k , $k = 1, 2, \dots, b$ or until σ is less than the specified tolerance. If the defined basis contains two or more linearly dependent functions, or one or more basis functions equal zero, then execution is terminated (the normal system is singular). The basis must in this case be changed.

SLNLLS.

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

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

be the function that defines the difference squared of the set of data to the model $f(a_1, a_2, \dots, a_K, X)$, one determines $a_1, a_2, \dots, a_K$ so that the function f is minimized. $f(a_1, a_2, \dots, a_K, X)$ is defined by the user in the calling program as FNF(X). Let C_r , $r = 1, 2, \dots, n$ 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_r^1 , $r = 1, 2, \dots, n$ be given. Calculate λ_n^0 to minimize

$$f(p_n^0 + \lambda_n^0 \xi_n^1)$$

and let

$$p_{n+1}^0 = p_n^0 + \lambda_n^0 \xi_n^1$$

let $t = 1$ and commence iteration with $K = 1$

Iteration

$$K: p_{n+1}^{K-1}, \xi_r^K, r = 1, 2, \dots, n$$

and t are given.

Step 1: find α to minimize

$$f(p_{n+1}^{K-1} + \alpha C_t)$$

CALCULATION METHODS

Update by

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

If $\alpha \neq 0$, let $p_0^K = p_{n+1}^{K-1} + \alpha C_t$.

If $\alpha = 0$, repeat step 1.

Should step 1 be repeated n times in succession then the point p_{n+1}^{K-1} is optimal.

Step 2: For $r = 1, \dots, n$ 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_{n+1}^K to minimize $f(p_n^K + \lambda_{n+1}^K \xi_{n+1}^K)$

and set $p_{n+1}^K = p_n^K + \lambda_{n+1}^K \xi_{n+1}^K$

Define $\xi_r^{K-1} = \xi_{r+1}^K \quad r = 1, \dots, n$

and continue with next iteration.

Some discussion of the procedure may be in order.

Step 1 proceeds cyclically through the coordinate directions. That is, each time we return to step 1 we use the next coordinate direction, repeating C_1 after using C_n . Every $n + 1$ times step 1 is employed the same coordinate direction is employed. The t indexes the coordinate direction to be used. If step 1 is repeated n times in succession, then all n coordinate directions 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 f is assumed strictly convex and continuously differentiable, that point is optimal.

In general step 1 is repeated until a new point is generated. In step 2 the procedure continues as in the earlier procedures. It is important to observe that after at most n iterations, all coordinate directions have been used. Both in step 1 (where we check if the found point is a minimum) and in step 2; it is necessary to minimize the function along a line. To find the minimum on a line, the following data must be provided:

1. A point on the line, p .
2. The direction of the line, ξ .
3. An upper bound to the length of step along the line, m .
4. The length of step along the line, $m/10$.
5. 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 $f(p+q\xi)$ are calculated, and then either $f(p-q\xi)$ or $f(p+2q\xi)$ is evaluated depending on whether $f(p)$ is less than or greater than $f(p+q\xi)$.

These three function values are now used in the general formula which predicts the turning value of the quadratic defined by a , $f(p+a\xi)$, b , $f(p+b\xi)$, c and $f(p+c\xi)$ to be at $(p+d\xi)$, 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 $f(p+d\xi)$ 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 which is furthest from $(p+d\xi)$ is discarded, so that the prediction may be repeated. Otherwise d is compared with a , b and c , if it is within the required accuracy of one of them, that point is chosen as the minimum. If not, $f(p+d\xi)$ is calculated so that the quadratic prediction may be repeated; the function value which is discarded out of $f(p-a\xi)$, $f(p+b\xi)$ and $f(p+c\xi)$ is normally the greatest, but only if rejecting a smaller one cannot yield a definite bracket on a minimum, which would not be otherwise obtained.

SLFSYT.

The Forsythe polynomials are a set of orthogonal polynomials over a discrete set of N weighted data points (X_i, Y_i, W_i) $1 \leq i \leq N$. The algorithm for generating these polynomials is as follows

$$P_{-1}(x) = 0$$

$$P_0(x) = 1$$

$$P_1(x) = (x - U_1) P_0(x)$$

$$P_2(x) = (x - U_2) P_1(x) - V_1 P_0(x)$$

$$\vdots$$

$$P_m(x) = (x - U_m) P_{m-1}(x) - V_m P_{m-2}(x)$$

where

$$U_m = \frac{\sum_{i=1}^N x_i [P_{m-1}(x_i)]^2 w_i}{D_{m-1}}$$

$$V_{m-1} = \frac{\sum_{i=1}^N x_i P_{m-1}(x_i) P_{m-2}(x_i) w_i}{D_{m-2}}$$

$$D_m = \sum_{i=1}^N [P_m(x_i)]^2 w_i$$

The aim is to approximate the function $f(x)$ by a linear combination of the polynomials i.e.

$$y(x) = \sum_{m=0}^M a_m P_m(x)$$

$y(x)$ is the desired approximation. The coefficients are given by:

$$a_m = \frac{\sum_{i=1}^N w_i y_i P_m(x_i)}{D_m}$$

It is possible to rewrite the equation in terms of x as follows:

$$y(x) = \sum_{m=0}^M a_m P_m(x) = \sum_{K=0}^M c^K x^K$$

where

$$c^K = \sum_{m=K}^M a_m b_r^m$$

and

$$b_k^m = \begin{cases} 0 & k < 0, \text{ or } k > m \\ 1 & k = m \\ b_{K-1}^{m-1} - u_m b_k^{m-1} - v_{m-1} b_k^{m-x} & 0 \leq k < m \end{cases}$$

The standard deviation is given by

$$\sigma_m = \sqrt{\frac{\left(\sum_{i=1}^N y_i^2\right) - \frac{\sum_{k=1}^m a_k^2 D_k}{\sum_{i=1}^N w(x_i)}}{\sum_{i=1}^N w(x_i)}}$$

If σ_m becomes less than a specified tolerance or $|\sigma_{m+1} - \sigma_m| < 10^{-6}$ then the routine terminates.

The subroutine computes the coefficients C_K .

This polynomial may then be evaluated using SLPLRR.

SLFTRP.

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

$$x_0, x_0 + h, x_0 + 2h, \dots$$

The transformation:

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

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

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

where

$0 \leq M \leq N-1$ for an even number of observations and

$0 \leq M \leq N$ for an 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 and b coefficients are calculated using Goertzel's algorithm. It is necessary to distinguish between two cases:

1. Odd number of observations (2N+1)

Let:

$$C_0 = 1$$

$$S_0 = 0$$

$$C_1 = \cos\left(\frac{2\pi}{2N+1}\right) \quad S_1 = \sin\left(\frac{2\pi}{2N+1}\right)$$

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

(Law of cosines)

(Law of sines)

The routine calculates:

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

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

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

then:

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

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

2. Even number of observations (2N)

Let:

$$C_0 = 1$$

$$S_0 = 0$$

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

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

(Law of cosines)

(Law of sines)

The routine calculates:

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

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

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

then:

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

$$b_K = \frac{1}{N} S_K U_1 \quad K = 0, 1, \dots, N$$

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

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

By observing the behaviour of $\bar{E}_m$ 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.

SLFOUI. may then be used for interpolation using the found coefficients.

SLFOUI.

Given the coefficients of the Fourier expansion approximating a function (see SLFTRP.), this expansion is evaluated for a given argument. The argument may be entered in radians, degrees, original units or with unit spacing. The relationship between the various scales may be seen as follows:

1. Odd number ($2N+1$) of observations over $(0, 2\pi)$

original scale:



where h = space between the observations (see SLFTRP)

unit spacing:



radians:



degrees:

$$\vartheta: \begin{array}{ccccc} & \cdot & & \cdot & \\ & 360 & & 2(360) & \\ 0 & \frac{\quad}{2N+1} & & \frac{\quad}{2N+1} & \\ & & & & \frac{2N \cdot 360}{2N+1} \\ & & & & 360 \end{array}$$

2. Even number ($2N$) of observations over $(0, 2\pi)$

original scale:

$$x \quad x_0 \quad x_0 + h \quad x_0 + 2h \quad \dots \quad x_0 + (2N - 1)h \quad x_0 + 2Nh$$

unit spacing:

radians:

$$\alpha \quad \cdot \quad \frac{\pi}{N} \quad \frac{2 \pi}{N} \quad \frac{(2 N-1) \pi}{N} \quad 2 \pi$$

degrees:

$$\vartheta \begin{array}{ccccccc} & & & & & & \\ & & & & & & \\ & & & & & & \\ 0 & \frac{360}{2N} & \frac{2(360)}{2N} & & \frac{(2N-1)360}{2N} & & 360 \end{array}$$

SLPLYF.

Let

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0 \quad (1)$$

The Lagrange interpolating polynomial $p(x)$ is the following:

$$\begin{aligned}
 p(x) = & \frac{(x - x_2) (x - x_3) \cdot \cdot \cdot (x - x_n)}{(x_1 - x_2) (x_1 - x_3) \cdot \cdot \cdot (x_1 - x_n)} \cdot y_1 \\
 & + \frac{(x - x_2) (x - x_3) \cdot \cdot \cdot (x - x_n)}{(x_2 - x_1) (x_2 - x_3) \cdot \cdot \cdot (x_2 - x_n)} \cdot y_2 \\
 & \quad \vdots \\
 & + \frac{(x - x_1) (x - x_2) (x - x_{n-1})}{(x_n - x_1) (x_n - x_2) \cdot \cdot \cdot (x_n - x_{n-1})} \cdot y_n
 \end{aligned} \tag{2}$$

Each term on the right-hand 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) becomes:

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

Let

$$\prod_{\substack{j=1 \\ j \neq i}}^n (x - x_j) = b_{i, n-1} x^{n-1} + b_{i, n-2} x^{n-2} + \dots + b_{i,1} x + b_{i,0}$$

these b_{ij} can be computed in a recursive fashion by successively multiplying one or more terms. In particular, suppose for simplicity $i > k+1$ and $1 < k+1 < n$, then:

$$B_{n-1} = B_{n-2} = \dots = B_{k+1} = \emptyset, B_k, \dots, B_1, B_0$$

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:

$$B'_{n-1} = 0$$

$$B'_{n-2} = 0$$

$$\vdots$$

$$B'_{k+1} = B_k$$

$$B'_k = B_{k-1} - x_{k+1} B_k$$

$$B'_{k-1} = B_{k-2} - x_{k+1} B_{k-1}$$

$$\vdots$$

$$B'_1 = B_0 - x_{k+1} B_1$$

$$B'_0 = -x_{K-1} B_0$$

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

$$a_j = \sum_{i=1}^n \frac{b_{ij}}{A_i} y_i \quad 0 \leq i \leq n-1.$$

This polynomial may then be evaluated using SLPLRR., remembering that evaluation is strictly valid only in the interval defined by the n base points, $x_1, \dots, x_n$.

SLCSPC.

Given the set of data (x_i, y_i) $i = 1, 2, \dots, N$ 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}) :

$$f_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

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 $i = 1, 2, \dots, (N-1)$.

The function $f(x)$ on the entire interval x, x_N consists of the 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 2nd derivative only.

The method used to determine these coefficients is due to Akima.

The cubic spline may be evaluated using SLCSPI.

SLCSPI.

For discussion of how to generate the interpolatory cubics see SLCSPC.

SLPADE.

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 $x = x_0$, where x_0 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_0 and so do the first $(m+n)$ derivatives.

The coefficients of the Taylor series expansion of $f(x)$ centered at x_0 will be used to find $P_m(x)$ and $Q_n(x)$. Since the polynomials depend only on the Taylor Series coefficients and not directly on the value x_0 , a change of variable making $x_0 = 0$ will have the effect that the a 's and the b 's may be computed directly from the coefficients.

With the necessary assumptions of convergence, let $\sum_{j=0}^{\infty} c_j (x - x_0)^j$

be the Taylor series expansion for $f(x)$:

$$P_m(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m = \sum_{j=0}^m a_j x^j$$

and

$$Q_n(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n = \sum_{j=0}^n b_j x^j$$

so that

$$\frac{P_m(x)}{Q_n(x)} \text{ approximates } \sum_{j=0}^{\infty} c_j x^j \text{ near } x = 0$$

$$\frac{P_m(x - x_0)}{Q_n(x - x_0)} \text{ will approximate } f(x) = \sum_{j=0}^{\infty} c_j (x - x_0)^j \text{ near } x = x_0$$

Since it is possible to divide both P and Q by a common factor, assume that they have no common factor and also that $b_0 = 1$. To approximate

$$\sum_{j=0}^{\infty} c_j x^j \text{ by } \frac{P_m(x)}{Q_n(x)} \text{ near } x = 0,$$

consider

$$E(x) = \sum_{j=0}^{\infty} c_j x^j - \frac{P_m(x)}{Q_n(x)},$$

and choose the a's and b's 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.

$$E(x) = \sum_{j=0}^{\infty} c_j x^j - \frac{P_m(x)}{Q_n(x)} = \frac{\left(\sum_{j=0}^{\infty} c_j x^j\right) \left(\sum_{j=0}^n b_j x^j\right) - \sum_{j=0}^m a_j x^j}{\sum_{j=0}^n b_j x^j}$$

$$= \frac{\sum_{j=0}^{\infty} \sum_{k=0}^j c_{j-k} b_k x^j - \sum_{j=0}^m a_j x^j}{\sum_{j=0}^n b_j x^j}$$

$$= \frac{\sum_{j=0}^m \sum_{k=0}^j (c_{j-k} b_k - a_j) x^j + \sum_{j=m+1}^{\infty} \sum_{k=0}^j c_{j-k} b_k x^j}{\sum_{j=0}^n b_j x^j}$$

Since the numerator and denominator are assumed to have no common factor, the vanishing of the first $(m+n)$ derivatives of $E(x)$ at 0 is equivalent to the vanishing of the first $(m+n+1)$ terms of the numerator:

$$\sum_{K=0}^j (c_{j-K} b_K - a_j) = 0 \quad \text{for } 0 \leq j \leq m \quad (1)$$

$$\sum_{K=0}^j c_{j-K} b_K = 0 \quad \text{for } m+1 \leq j \leq m+n \quad (2)$$

The set (2) of n equations may be solved for the n unknown b's. Then the set (1) of equations can be used to compute the a's:

$$a_j = \sum_{K=0}^j c_{j-K} b_K$$

The accuracy of the approximation improves as the sum of the degrees of the numerator and denominator increases. Also the best approximations occur when the degrees of the numerator and denominator are nearly equal.

The user may supply either the Taylor coefficients or the derivatives at the centre of the interval. In the latter case, the routine calculates the corresponding Taylor coefficients.

The polynomials may be evaluated using SLPLRR. with argument $x = x_0$.

NUMERICAL INTEGRATION AND DIFFERENTIATION

SLITEQ.

A Newton-Cotes interpolatory method for orders 1 to 8 is used. An order p quadrature fits a polynomial of degree p to $p+1$ points of the data base. This polynomial is then integrated. The formulae for each quadrature and truncation error term are given below.

$$P = 1 \int_{x_{i-1}}^{x_i} f(x) dx = \frac{h}{2} (f_i + f_{i-1}) - \frac{h^3}{12} f^{(2)}(\xi) \quad x_{i-1} \leq \xi \leq x_i$$

$$P = 2 \int_{x_{i-2}}^{x_i} f(x) dx = \frac{h}{3} (f_i + 4f_{i-1} + f_{i-2}) - \frac{h^5}{90} f^{(4)}(\xi) \quad x_{i-2} \leq \xi \leq x_i$$

$$P = 3 \int_{x_{i-3}}^{x_i} f(x) dx = \frac{3h}{8} (f_i + 3f_{i-1} + 3f_{i-2} + f_{i-3}) - \frac{3h^5 f^{(4)}(\xi)}{80} \quad x_{i-3} \leq \xi \leq x_i$$

$$P = 4 \int_{x_{i-4}}^{x_i} f(x) dx = \frac{2h}{45} (7f_i + 32f_{i-1} + 12f_{i-2} + 32f_{i-3} + 7f_{i-4}) - \frac{8}{945} h^7 f^{(6)}(\xi) \quad x_{i-4} \leq \xi \leq x_i$$

$$P = 5 \int_{x_{i-5}}^{x_i} f(x) dx = \frac{5h}{288} (19f_i + 75f_{i-1} + 50f_{i-2} + 59f_{i-3} + 75f_{i-4} + 19f_{i-5}) - \frac{275}{12096} h^7 f^{(6)}(\xi) \quad x_{i-5} \leq \xi \leq x_i$$

$$P = 6 \int_{x_{i-6}}^{x_i} f(x) dx = \frac{h}{140} (41f_i + 216f_{i-1} + 27f_{i-2} + 272f_{i-3} + 27f_{i-4} + 216f_{i-5} + 41f_{i-6}) - \frac{9}{1400} h^9 f^{(8)}(\xi) \quad x_{i-6} \leq \xi \leq x_i$$

$$P = 7 \int_{x_{i-7}}^{x_i} f(x) dx = \frac{7h}{17280} (751f_i + 3577f_{i-2} + 2989f_{i-3} + 2989f_{i-4} + 1323f_{i-5} + 3577f_{i-6} + 751f_{i-7}) - \frac{8183}{518400} h^9 f^{(8)}(\xi) \quad x_{i-7} \leq \xi \leq x_i$$

$$P = 8 \int_{x_{i-8}}^{x_i} f(x) dx = \frac{4h}{14175} (989f_i + 5888f_{i-1} - 928f_{i-2} + 10496f_{i-3} - 4540f_{i-4} + 10496f_{i-5} - 928f_{i-6} - 5888f_{i-7} + 989f_{i-8}) - \alpha h^{11} f^{(16)}(\xi) \quad x_{i-8} \leq \xi \leq x_i$$

SLDTEQ.

Let (x_i, y_i) $i = 1, \dots, N$, $N \geq 6$, be the data table. For each point, 4 other points in the neighbourhood are selected. A 5 point difference formula is used for both the first and second derivative.

Forward Difference (For $i = 1$ and 2)

$$f'(x_i) = \frac{1}{12h} [-25f(x_i) + 48f(x_i + h) - 36f(x_i + 2h) + 16f(x_i + 3h) - 3f(x_i + 4h)]$$

ε_T (truncation error) is proportional to $h^4 f^{(5)}(\xi)$

$x_i \leq \xi \leq x_i + 4h$

$$f''(x_i) = \frac{1}{12h^2} [35f(x_i) - 104f(x_i + h) + 114f(x_i + 2h) - 56f(x_i + 3h) + 11f(x_i + 4h)]$$

ε_T (truncation error) is proportional to $h^3 f^{(5)}(\xi)$

$x_i \leq \xi \leq x_i + 4h$

Central Difference (For $i = 3, \dots, N-2$)

$$f'(x_i) = \frac{1}{12h} [f(x_i - 2h) - 8f(x_i - h) + 8f(x_i + h) - f(x_i + 2h)]$$

ε_T (truncation error) is proportional to $h^4 f^{(5)}(\xi)$

$x_i - 2h \leq \xi \leq x_i + 2h$

$$f''(x_i) = \frac{1}{12h^2} [-f(x_i - 2h) + 16f(x_i - h) - 30f(x_i) + 16f(x_i + h) - f(x_i + 2h)]$$

ε_T (truncation error) is proportional to $h^3 f^{(5)}(\xi)$

$x_i - 2h \leq \xi \leq x_i + 2h$

Backward Difference

(For $i = N-1, N$)

$$f'(x_i) = \frac{1}{12h} [3f(x_i - 4h) - 16f(x_i - 3h) + 36f(x_i - 2h) - 48f(x_i - h) + 25f(x_i)]$$

ε_T (truncation error) is proportional to $h^4 f^{(5)}(\xi)$

$$x_i - 4h \leq \xi \leq x_i$$

$$f''(x_i) = \frac{1}{12h^2} [11f(x_i - 4h) - 56f(x_i - 3h) + 114f(x_i - 2h) - 104f(x_i - h) + 35f(x_i)]$$

ε_T (truncation error) is proportional to $h^3 f^{(5)}(\xi)$

$$x_i - 4h \leq \xi \leq x_i$$

These formulae are exact for polynomials of degree less than or equal to 4.

SLDIFF.

The first and second derivatives are computed using a 5 point central difference formula.

$$f'(x_0) = \frac{1}{12h} [f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)]$$

ε_T (truncation error) is proportional to $h^4 f^{(5)}(\xi)$

$$x_0 - 2h \leq \xi \leq x_0 + 2h$$

and

$$f''(x_0) = \frac{1}{12h^2} [-f(x_0 - 2h) + 16(f(x_0 - h) + f(x_0 + h)) - 30f(x_0) - f(x_0 + 2h)]$$

ε_T (truncation error) is proportional to $h^3 f^{(5)}(\xi)$

$$x_0 - 2h \leq \xi \leq x_0 + 2h$$

The increment h is set by the user and the following will give accurate results:

$h \geq 0.005$ to evaluate the 1st derivative only

$h \geq 0.06$ (minimum) to evaluate both derivatives.

SLROMB.

Romberg quadrature is a combination of the composite trapezoidal rule for successively smaller mesh size and a Richardson type extrapolation.

A triangular scheme is generated

$$T_1^1 \quad T_1^2 \quad T_1^3 \quad T_1^4 \dots$$

$$T_2^1 \quad T_2^2 \quad T_2^3 \dots$$

$$T_3^1 \quad T_3^2 \dots$$

The first row is the result of the composite trapezoidal rule for successively finer meshes (i.e. more subdivisions). The other elements of the scheme are the result of extrapolating the two entries directly above. In particular,

$$T_I^k = \frac{4^{I-1} T_{I-1}^{(k+1)} - T_{I-1}^{(k)}}{4^{I-1} - 1} \quad \begin{array}{l} I = 2, 3, \dots \\ k = 1, 2, \dots \end{array}$$

The scheme is computed from the left hand corner downwards. The process stops if, for some k

$$1. \quad T_{k+1}^i - T_k^i < E$$

for some tolerance E , or

$$2. \quad T_{k+1}^1 - T_k^1 / T_{k+1}^1 < E$$

Otherwise, the program halts after a specified number of lines have been calculated.

SLGAUS.

Let

$$I = \int_A^B f(x) dx$$

be the integral to be evaluated by an order n quadrature. A transformation of variables is performed

$$X = \frac{B-A}{2} \xi + \frac{B+A}{2}$$

Then

$$\int_{-1}^1 f(\xi) d\xi = \sum_{i=1}^n W_i f(\xi_i)$$

that is, a weighted sum of function values at the zero ξ_i of the Legendre

polynomials $P_n(x)$ of order n , with weights

$$W_i = \frac{2}{(1 - \xi_i) [P_n(\xi_i)]^2}$$

The zeros and weights are stored on the external sequential data file WEIGH1.

SLLAGU.

For an order n quadrature

$$\int_0^\infty e^{-x} f(x) dx = \sum_{i=1}^n w_i f(\xi_i)$$

and

$$\int_0^\infty f(x) dx = \sum_{i=1}^n W'_i f(\xi_i)$$

where

$$W'_i = e^{-\xi_i} w_i$$

i.e., a weighted sum of function values at the zeros ξ_i of the Laguerre polynomials $L_n(x)$ of degree n and with weights,

$$w_i = \frac{-(n-1)!^2}{L_n(\xi_i) L_{n-1}(\xi_i)}$$

The zeros ξ_i and weights w_i and w'_i are stored on the external data file WEIGH2.

ORDINARY DIFFERENTIAL EQUATIONS

SLEROM.

An N -th order equation can be rewritten as a system of N first order ordinary differential equations by using the substitution:

$$y_1 = y, y_2 = \frac{dy}{dx}, \dots, y_{N-1} = \frac{d^{N-2} y}{dx^{N-2}}, y_N = \frac{d^{N-1} y}{dx^{N-1}}$$

This transformation is automatically performed by the routine. The

following algorithm can then be used for either a system of first order equations or a single, higher order equation. The case of one equation on $[x_i, x_{i+1}]$ is described, i.e.

$$y' = f(x, y) \quad ; \quad y_0 = y(x_0)$$

The modified Euler or improved polygon method is used as a quickly computed first approximation using step size $h = x_{i+1} - x_i$

$$\bar{y}_{i+1} = y_i + h f(x_i + h/2, \bar{y}_{i+1/2})$$

$$\bar{y}_{i+1/2} = y_i + \frac{h}{2} f(x_i, y_i)$$

The step from x_i to x_{i+1} is taken using successively smaller step sizes i.e. $h, h/2, h/4, h/6, \dots$. An extrapolation to zero step size is performed at each stage yielding the triangular array:

$$\begin{array}{cccc} A_{0,0} & & & \\ A_{1,0} & A_{1,1} & & \\ A_{2,0} & A_{2,1} & A_{2,2} & \\ \vdots & \vdots & \vdots & \\ \vdots & \vdots & \vdots & \end{array}$$

where

$$A_{m,0} = y\left(x_i + \frac{h}{2^m}\right) \quad m = 0, 1, \dots$$

$$A_{m,n+1} = \frac{2^{p+n} A_{m,n} - A_{m-1,n}}{2^{p+n} - 1} \quad n = 0, 1, \dots, m-1$$

where p = order of the method used to compute $A_{m,0}$. In this case $p = 2$. The doubling and halving of the step size is decided by the measure of convergence of this extrapolation to a solution.

$$|A_{j+1,j+1} - A_{j,j}| / |A_{j,j}|$$

is the measure. If this term is larger than a pre-assigned tolerance, the step size is halved and the step is repeated. If this term is smaller than the tolerance by a factor of .01 sufficiently often, the step size is doubled. At most six lines of the triangular array are generated. The tolerance and the maximum number of halvings are function parameters.

”

”

”

”

”

B. SUBROUTINES AND RELATED TEST PROGRAMS

ABOUT THIS CHAPTER

This chapter contains a table of the Subroutines and their related Test Programs.

CONTENTS

SUBROUTINES AND RELATED TEST PROGRAMS	B-1
--	-----

SUBROUTINES AND RELATED TEST PROGRAMS

NAME	ADDRESS	TEST PROGRAM
SLCONN.	30401	SLCONN
SLPRIM.	30601	SLPRIM
SLEUCL.	30801	SLEUCL
SLRFCO.	31001	SLRFCO
SLSUCO.	31201	SLSUCO
SLCFCO.	31401	SLCFCO
SLFACT.	31601	SLFACT
SLBINO.	31801	SLBINO
SLMULT.	32001	SLMULT
SLDUPL.	32201	SLDUPL
SLATN2.	32401	SLATN2
SLCONV.	32601	SLCONV
SLRPCC.	32801	SLRPCC
SLPRCC.	33001	SLPRCC
SLCSIN.	33201	SLCSIN
SLCCOS.	33401	SLCCOS
SLCTAN.	33601	SLCATN
SLCASN.	33801	SLCASN
SLCACS.	34001	SLCACS
SLCATN.	34201	SLCATN
SLCLN.	34401	SLCLN
SLCEXP.	34601	SLCEXP
SLCRZ.	34801	SLCRZ
SLCZMZ.	35001	SLCZMZ
SLCZDZ.	35201	SLCZDZ
SLCSQR.	35401	SLCSQR
SLCZN.	35601	SLCZN
SLCZA.	35801	SLCZA
SLPLRC.	36001	SLPLRC
SLPLRR.	36201	SLPLRR
SLPRRR.	36401	SLPRRR
SLPLYM.	36601	SLPLYM
SLPLYD.	36801	SLPLYD
SLPTRA.	37001	SLPTRA
SLKMF.	37201	SLKMF
SLEMF.	37401	SLEMF
SLLAGG.	37601	SLLAGG
SLHNF.	37801	SLHNF
SLHEN.	38001	SLHEN
SLFOUR.	38201	SLFOUR
SLGAMA.	38401	SLGAMA
SLERF.	38601	SLERF
SLBJN.	38801	BJN

NAME	ADDRESS	TEST PROGRAM
SLIIØX.	39001	INT
SLSF.	39201	SFR
SLCF.	39401	CFR
SLCHYF.	39601	CHYF
SLGHYP.	39801	GHYP
SLSIF.	40001	SIF
SLCINF.	40201	CINF
SLEIF.	40401	EIF
SLEINF.	47001	EINF
SLBAIR.	40601	BAIR
SLRBIS.	40801	BISEL
SLNLIN.	41001	NLIN
SLCRO.	41201	SLCRO2
SLCHO.	41401	SLCHO2
SLJAC.	41601	SLJAC2
SLHES.	41801	SLHES2
SLHEV.	46001	SLHES2
SLCHA.	42201	SLCHA2
SLJCB.	42401	SLJCB2
SLLLSQ.	42601	RRLLSQ
SLNLLS.	46401	RRNLLA
SLFSYT.	43001	RRFSYT
SLFTRP.	43201	RRFTRP
SLFOUI.	43401	RRFTRP
SLPLYF.	43601	RRPLYF
SLCSPC.	43801	RRSPC
SLCSPI.	44001	RRSPC
SLPADE.	44201	RRPADE
SLITEQ.	44401	SLITEQ
SLDTEQ.	44601	SLDTEQ
SLDIFF.	44801	SLDIFF
SLROMB.	45001	SLROMB
SLGAUS.	45201	SLGAUS
SLLAGU.	45401	SLLAGU
SLEROM	45601	SLEROM

NOTICE

Ing. C. Olivetti & C. S.p.A. reserves the right to make improvements in the product described in this manual at any time and without notice.

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 notwithstanding, all software being licensed to Customer is licensed "as is". THERE ARE NO WARRANTIES EXPRESS OR IMPLIED INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTY OF FITNESS FOR PURPOSE AND OLIVETTI SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES IN CONNECTION WITH SUCH SOFTWARE.

The enclosed programs are protected by Copyright and may be used only by the Customer. Copying for use by third parties without the express written consent of Olivetti is prohibited.

GU Code 3987570 J (0)

Printed in Italy



GU Code 3987570 J (0)

Drawn No. - - - - -

Printed in Italy