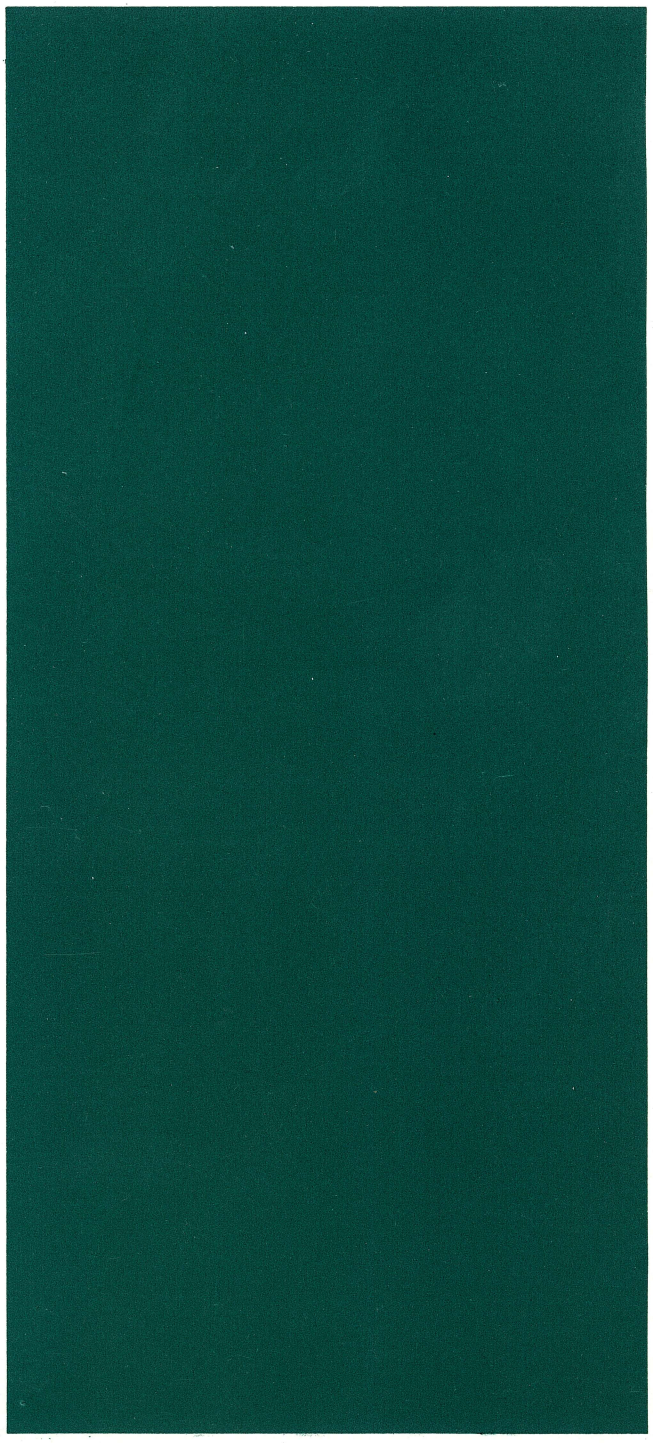
Honeywell Bull

TIME-SHARING APPLICATIONS LIBRARY GUIDE VOLUME I - MATHEMATICS

SERIES 6000/600

APPLICATIONS



Ref. : 19.53.106 A

Honeywell Bull

SERIES 6000/600

TIME-SHARING APPLICATIONS LIBRARY GUIDE VOLUME I—MATHEMATICS

SUBJECT:

Descriptions, Sample Problems, and Solutions of Mathematical Time-Sharing Programs.

SPECIAL INSTRUCTIONS:

This manual, in conjunction with *Series 6000/600 Time-Sharing Applications Library Guide Volume II—Statistics* (Order No. DA44) and *Volume III—Industry* (Order No. DA45), supersedes *GE-600 Line Time-Sharing Library Programs*, Document No. CPB-1694. The contents of CPB-1694 have been divided into three volumes. Sixty-five programs have been added to the new set of manuals. They are listed on the back of this page.

DATE:

June, 1971

ORDER NUMBER: DA43

Rev. 0

Printed in France

Ref.: 19.53.106 A

ADDITIONAL PROGRAMS INCORPORATED IN THE MAY 1971 EDITIONS

Series 6000/600 Time-Sharing
Applications Library Guide
Volume I - Mathematics,
Order No. DA43

4SQRS
ARCTAN
COMP1
EIGSR
EUALG
FRESNL
GAHER
GALA
GAUSSN
GCDN
JACELF
LINSR
ORTHP
POLYC
QUADEQ
ROMBINT
STIRLING
ZCOP
ZCOP2
ZORP2

Series 6000/600 Time-Sharing
Applications Library Guide
Volume II - Statistics,
Order No. DA44

ANOVA
ANVA3
FACTAN
FLAT
FORIR
LINREG
MREG1
POISON
POLFT
PROBC
RANDX
RNDNRM
STAT01
STAT02
STAT04
STAT05
STAT06
STAT08
STAT09
STAT11
STAT12
STAT13
STAT14
STAT15
STAT16
STAT18
STAT33
UNIFM
URAN
XNOR1
XNORM

Series 6000/600 Time-Sharing
Applications Library Guide
Volume III - Industry,
Order No. DA45

BUSINESS AND FINANCE
- DEPREC
- RETURN

MANAGEMENT SCIENCE AND OPTIMIZATION
- GASPIIA
- PERT
- SMOOTH
- TCAST

ENGINEERING
- ACNET
- NLNET
- PVT

GEOMETRIC AND PLOTTING
- PLOT

EDUCATION AND TUTORIAL
- DRIVES
- EXPERN
- PREPRS

UTILITY AND MISCELLANEOUS
- CONVRT

PREFACE

This manual describes and tells how to use the mathematical time-sharing programs available with the Series 6000 and 600 information processing systems. The programs are listed alphabetically in the Table of Contents.

The writeup about each program includes the purpose of the program; the language in which the program is written; the method of approach, if applicable; instructions for its use; restrictions of the program, if any; and sample problems and their solution. In the sample solutions, all information that the user types is underlined.

The instructions in this manual assume that the programs are available in the user master catalog LIBRARY, and are accessible with READ or EXECUTE permission. In the sample solution printouts the programs had already been accessed using the GET command, and/or copied onto the current file using the OLD or LIB command.

Time-sharing programs for statistics and other classifications are also available. Individual manuals are published for these categories as follows:

Series 6000/600 Time-Sharing Applications Library Guide Volume II - Statistics, Order No. DA44

Series 6000/600 Time-Sharing Applications Library Guide Volume III - Industry, Order No. DA45

The Industry manual is organized into sections by type as follows:

BF - Business and Finance

MS - Management Science and Optimization

EN - Engineering

GP - Geometric and Plotting

ED - Education and Tutorial

DE - Demonstration

UM - Utility and Miscellaneous

Each section is paginated with the 2-letter identifier that is shown above.

A complete listing of the programs in the library is available by listing the LIBRARY program CATALOG. A copy of this program follows the Table of Contents for your information.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

CATALØG ØF SERIES 6000/600 T-S LIBRARY PRØGRAMS

FØRMAT INDICATØR:

FIRST LETTER	FØLLØWING LETTERS	
F FØRTRAN-SØURCE	P	(ØR BLANK) PRØGRAM
Ø FØRTRAN-ØBJECT	S	SUBRØUTINE(S)
C CARDIN	F	FUNCTIØN(S)
B BASIC-SØURCE	P-S	PRØGRAM WITH EXTRACTABLE SUBRØUTINE
E EDITØR(ASCII)		

SUBJECTS	DØCUMENTATIØN MANUAL
MATHEMATICS (MA)	ØRDER # DA43
INTEGRATIØN	
DIFFERENTIATIØN, DIFFERENTIAL EQ.	
INTERPØLATIØN	
PØLYNØMIALS	
LINEAR EQUATIØNS	
MATRICES	
NØN-LINEAR EQUATIØNS	
SPECIAL FUNCTIØN EVALUATIØN	
LØGIC AND NUMBER THEØRY	
STATISTICS (ST)	ØRDER # DA44
CURVE FITTING AND REGRESSIØN	
ANALYSIS ØF VARIANCE	
PRØBABILITY DISTRIBUTIØNS	
CØNFIDENCE LIMITS	
HYPØTHESIS TESTING	
DESCRIPTIVE STATISTICS	
RANDØM NUMBER GENERATIØN	
MISCELLANEØUS STATISTICS	
BUSINESS AND FINANCE (BF)	ØRDER # DA45
MANAGEMENT SCIENCE AND ØPTIMIZATIØN (MS)	
LINEAR PRØGRAMMING	
INTEGER PRØGRAMING	
NØN-LINEAR ØPTIMIZATIØN	
NETWØRK ANALYSIS	
FØRECASTING	
SIMULATIØN	
ENGINEERING (EN)	
GEØMETRIC AND PLØTTING (GP)	
EDUCATIØN AND TUTØRIAL (ED)	
DEMØNSTRATIØN (DE)	
UTILITY AND MISCELLANEØUS (UM)	

THE DØCUMENTATIØN FØR THESE PRØGRAMS IS AVAILABLE IN THREE MANUALS:
SEE ØRDER # DA43 FØR PRØGRAMS IN MATHEMATICS
ØRDER # DA44 FØR PRØGRAMS IN STATISTICS
ØRDER # DA45 FØR PRØGRAMS IN ALL ØTHER CATEGØRIES.

SUBRØUTINES THAT ARE CALLED BY A PRØGRAM AND MUST BE EXECUTED WITH IT ARE LISTED IN BRACKETS AT THE END ØF THE DESCRIPTIØN.

THESE PRØGRAMS HAVE ALL BEEN REVIEWED AND TESTED BUT NØ RESPØNSIBILITY CAN BE ASSUMED.

```
**********MA--MATHEMATICS**********

***INTEGRATION***
CLCINT      FF     INTEGRATION BY SIMPSON'S RULE
FINT        FF     EVALUATE FOURIER INTEGRALS BY FILON'S FORMULA
GAHER       FF     GAUSS-HERMITE QUADRATURE
GALA        FF     GAUSS-LAGUERRE QUADRATURE
GAUSSN      FF     EVALUATE DEFINITE DOUBLE OR TRIPLE INTEGRALS
GAUSSQ      FF     GAUSSIAN QUADRATURE
NCOATES     FP-S   NEWTON-COATES QUADRATURE
NUMINT      B      GAUSSIAN QUADRATURE
ROMBINT     FP-S   ROMBERG INTEGRATION

***DIFFERENTIATION, DIFFERENTIAL EQ.***
AMPBX       FS     ADAMS-MOULTON FOR 1ST-ORDER DIFF. EQNS [RKPBX]
FDRVUL      FF     DIFFERENTIATE TABULATED FUNCTION, UNEQUAL SPACING
HDRVEB      FF     DIFFERENTIATE TABULATED FUNCTION, EQUAL SPACING
RKPBX       FS     RUNGE-KUTTA FOR 1ST-ORDER DIFF. EQNS

***INTERPOLATION***
INT1        FF     SINGLE LAGRANGIAN INTERPOLATION [TLU1]
INT2        FF     DOUBLE LAGRANGIAN INTERPOLATION [TLU1]
INT2A       FF     VARIABLE DOUBLE LINEAR INTERPOLATION [TLU1]

***POLYNOMIALS***
BICOF       FS     CALCULATE BINOMIAL COEFFICIENTS
CLPLY       FF     EVALUATE REAL POLY AT REAL ARGUMENT
DVALG       FS     POLYNOMIAL DIVISION
EUALG       FS     G.C.D. OF TWO POLYNOMIALS [DVALG]
MTALG       FS     MULTIPLY POLYNOMIALS
PLMLT       FS     REAL POLY COEFFICIENTS RECONSTRUCTED FROM REAL ROOTS
POLRTS      FP     SOLUTION OF POLY BY BAIRSTOWS METHOD
POLYC       FS     REAL POLY COEFFICIENTS RECONSTRUCTED FROM COMPLEX ROOTS
POLYV       FS     EVALUATE REAL POLY AT COMPLEX ARGUMENT
QUADEQ      B      SOLUTION TO QUADRATIC EQUATIONS
ROOTER      B      SOLUTION OF POLY BY BAIRSTOWS METHOD
ZCOP        FP     ROOTS OF POLYNOMIAL WITH COMPLEX COEFF.
ZCOP2       FS     ROOTS OF POLYNOMIAL WITH COMPLEX COEF. [ZCOP2]
ZORP        FP     ROOTS OF REAL POLY
ZORP2       FS     ROOTS OF REAL POLY

***LINEAR EQUATIONS***
GJSIMEQ     FS     SOLVE LINEAR SYSTEMS BY GAUSS-JORDAN
GSEIDEL     FP-S   SOLVE LINEAR SYSTEMS BY GAUSS-SEIDEL
LINEQ       FS     SOLVE LINEAR SYSTEMS BY GAUSSIAN ELIMINATION
LINSR       FP     SOLVE LINEAR SYSTEMS BY GAUSSIAN ELIMINATION [LINEQ]
SIMEQN      B      SOLVE LINEAR SYSTEMS BY MATRIX INVERSION

***MATRICES***
DETE        FF     EVALUATE DETERMINANT OF REAL MATRIX
DOMEIG      FP-S   DOMINANT EIGENVALUES OF REAL MATRIX
EIG1        FS     EIGENVALUES OF SYM MATRIX BY JACOBI METHOD
EIGSR       FP     EIGENVALUES AND VECTORS OF REAL SYM. MATRIX [EIG1]
MTINV       FS     MATRIX INVERSION BY PIVOTS
MTMPY       FS     MATRIX MULTIPLICATION
SPEIG1      FS     SPECIAL EIGEN PROBLEMS [EIG1]
SYMEIG      FP     EIGENVALUES OF SYM MATRIX BY JACOBI METHOD

***NON-LINEAR EQUATIONS***
BROWN       FS     SOLN OF SIMULTANEOUS SYSTEMS BY BROWN METHOD
SECANT      FS     SOLN OF SIMULTANEOUS SYSTEMS BY SECANT METHOD [MTINV]
SOLN        FF     ZERO OF AN ARBITRARY FUNCTION
ZEROES      B      ZERO,MAX,MIN OF FUNCTION
```

```
***SPECIAL FUNCTION EVALUATION***
ARCTAN     FF     ARCTANGENT IN RADIANS OF Y/X
BESL       FS     BESSEL FUNCTION [GAMF]
COMP1      FF     EVALUATES REAL HYPERBOLIC TRIG FUNCTIONS
COMP2      FS     COMPLEX MULT. AND DIVISION
COMP3      FS     EVALUATES VARIOUS FUNCTIONS FOR COMPLEX ARGUMENT [COMP2]
ERRF       FF     ERROR FUNCTION
ERRINV     FF     INVERSE ERROR FUNCTION
FRESNL     FS     EVALUATES FRESNAL INTEGRALS
GAMF       FF     GAMMA FUNCTION
JACELF     FS     EVALUATES JACOBIAN ELLIPTIC FUNCTIONS SN, CN, DN
ORTHP      FF     EVALUATE ORTHOGONAL POLYNOMIALS
STIRLING   FP-S   N FACTORIAL BY STIRLINGS APPROXIMATION
TMFCEV     B      EVALUATE DAMPED OR UNDAMPED FOURIER SERIES

***LOGIC AND NUMBER THEORY***
4SQRS      B      WRITES INTEGERS AS SUM OF SQUARES OF FOUR INTEGERS
GCDN       FS     G.C.D. OF N INTEGERS


**********ST--STATISTICS**********

***CURVE FITTING AND REGRESSION***
CFIT       FP     LEAST SQRS. POLY. WITH RESTRAINTS
CURFIT     B      FITS SIX DIFFERENT CURVES BY LEAST SQRS
FORIR      FP     LEAST SQUARES ESTIMATE OF FINITE FOURIER SERIES MODEL
FOURIER    B      COEFF OF FOURIER SERIES TO APPROX A FUNCTION
LINEFIT    FS     LEAST SQRS LINE
LINREG     B      LST.SQRS. BY LINEAR, EXPONENTIAL, OR POWER FUNCTION
LSPCFP     FP     LEAST SQRS POLYNOMIAL FIT
LSQMM      FS     GENERALIZED POLY FIT BY LEAST SQRS OR MIN-MAX
MREG1      FP     MULTIPLE LINEAR REGRESSION
MULFIT     B      MULTIPLE LINEAR FIT WITH TRANSFORMATIONS
ORPOL      FP     LEAST SQRS FIT WITH ORTHOGONAL POLYS
POLFIT     B      LEAST SQRS POLYNOMIAL FIT
POLFT      FP     LEAST SQRS POLYNOMIAL FIT
SMLRP      FP     MULTIPLE LINEAR REGRESSION
SMLRPOBJ   O      OBJECT FILE FOR SMLRP

***ANALYSIS OF VARIANCE***
ANOVA      FP     ONE OR TWO WAY ANALYSIS OF VARIANCE
ANVA1      FP     ONEWAY ANALYSIS OF VARIANCE
ANVA3      FP     THREE WAY ANALYSIS OF VARIANCE
ANVA5      FP     MULTIPLE VARIANCE ANALYSIS
KRUWAL     FP     KRUSKAL-WALLIS 2-WAY VARIANCE [XINGAM]
ONEWAY     B      ONEWAY ANALYSIS OF VARIANCE
STAT13     B      ANALYSIS OF VARIANCE TABLE, 1-WAY RANDOM DESIGN
STAT14     B      ANALYSIS OF VARIANCE TABLE FOR RANDOMIZED BLOCK DESIGN
STAT15     B      ANALYSIS OF VARIANCE TABLE FOR SIMPLE LATIN-SQ DESIGN
STAT16     B      ANALYSIS OF VARIANCE TABLE, GRAECO-LATIN SQUARE DESIGN
STAT18     B      ANALYSIS OF VARIANCE TABLE, YOUDEN SQUARE DESIGN
STAT33     B      ANALYSIS OF VARIANCE TABLE, 1-WAY RANDOM DESIGN

***PROBABILITY DISTRIBUTIONS***
ANPF       FF     NORMAL PROBABILITY FUNCTION [ERRF]
BETA       FF     BETA DISTRIBUTION
BINDIS     B      BINOMIAL PROBABILITIES
EXPLIM     B      EXPONENTIAL DISTRIBUTIONS
POISON     FF     POISSON DISTRIBUTION FUNCTION
PROBC      FP     PROBABLITIES OF COMBINATIONS OF RANDOM VARIABLES
PROVAR     B      NORMAL AND T-DISTRIBUTION
TDIST      FF     T-DISTRIBUTION [BETA]
XINGAM     FF     INCOMPLETE GAMA FUNCTION
```

```
***CØNFIDENCE LIMITS***
BAYES      B      DIFFERENCE ØF MEANS IN NØN-EQUAL VARIANCE
BICØNF     B      CØNF. LIMITS FØR PØPULATIØN PRØPØRTIØN (BINØMIAL)
BINØM      FP     BINØMIAL PRØBABILITIES AND CØNFIDENCE BANDS
CØLINR     B      CØNFIDENCE LIMITS ØN LINEAR REGRESSIØNS
CØNBIN     B      CØNF. LIMITS FØR PØPULATIØN PRØPØRTIØN (NØRMAL)
CØNDIF     B      DIFFERENCE ØF MEANS IN EQUAL VARIANCE
CØNLIM     B      CØNF. LIMITS FØR A SAMPLE MEAN
STATØ5     B      CØNFIDENCE INTERVAL FØR MEAN BY SIGN TEST
STATØ6     B      CØNFIDENCE LIMITS,WILCØXØN SIGNED RANK SUM TEST

***HYPØTHESIS TESTING***
BITEST     B      TEST ØF BINØMIAL PRØPØRTIØNS
CHISQR     FS     CHI-SQUARE CALCULATIØNS
CØRREL     FP     CØNTINGENCY CØEFFICIENT [XINGAM]
CØRRL2     FP     CØRRELATIØN CØEFFICIENT [TDIST;BETA]
KØKØ       FP     KØLMØGØRØV-SMIRNØV TWØ SAMPLE TEST [XINGAM]
SEVPRØ     B      CHI-SQUARE
STATØ1     B      MEAN,STD ØF MEAN, ... , T-RATIØ,2 GRØUPS, PAIRED
STATØ2     B      MEANS, VARIANCES, A.D T-RATIØ 2 GRØUPS, UNPAIRED DATA
STATØ4     B      CHI-SQUARE AND PRØBABILITIES, 2X2 TABLES
STATØ8     B      CØMPARES TWØ GRØUPS ØF DATA USING THE MEDIAN TEST
STATØ9     B      CØMPARE 2 DATA GRØUPS,MANN-WHITNEY 2-SAMPLE RANK TEST
STAT11     B      SPEARMAN RANK CØRRELATIØN CØEF. FØR 2 SERIES ØF DATA
STAT12     B      CØMPUTES CØRRELATIØN MATRIX FØR N SERIES ØF DATA
TAU        FP     KENDALL-RANK CØRRELATIØN

***DESCRIPTIVE STATISTICS***
MANDSD     B      FIND MEAN, VARIANCE, STD
STAT       FP     FIND SEVERAL STATISTICS FØR SAMPLE DATA [ANFF;ERRF]
STATAN     B      FIND VARIØUS STATISTICAL MEASURES
TESTUD     B      SAMPLE STATISTICS
UNISTA     B      DESCRIPTIØN ØF UNI-VARIANT DATA

***RANDØM NUMBER GENERATIØN***
FLAT       ØF     UNIFØRM RANDØM NUMBER GENERATØR
FLATSØRC   C      CARDIN SØURCE FILE FØR FLAT
RANDX      FF     RANDØM #'S, UNIFØRM DIST. BETWEEN 0 AND 1
RNDNRM     FF     CALCULATES NØRMAL RANDØM NUM. [FLAT]
UNIFM      ØF     UNIFØRM RANDØM NUMBER GENERATØR
UNIFMSØR   C      CARDIN SØURCE FILE FØR UNIFM
URAN       ØF     UNIFØRM RANDØM NUMBER GENERATØR
URANSØRC   C      CARDIN SØURCE FILE FØR URAN
XNØR1      FF     NØRMAL RANDØM NUMBERS, VARIABLE MEAN, STD [RANDX]
XNØRM      FF     NØRMAL RANDØM NUMBERS, MEAN 0, STD 1. [RANDX]

***MISCELLANEØUS STATISTICS***
FACTAN     FP     FACTØR ANALYSIS
STADES     E      EXPLANATIØN ØF CØLINR,CURFIT,MULFIT,UNISTA


**********BF--BUSINESS AND FINANCE**********

ANNUIT     B      ANNUITIES,LØANS,MØRTGAGES
BLDGCØST   B      ANALYZE BUILDING CØSTS
DEPREC     B      CALCULATES DEPRECIATIØN BY FØUR METHØDS
SAVING     B      SAVINGS PLAN CALCULATIØNS
RETURN     B      CØMPUTES ANNUAL RETURNS FØR A SECURITY FRØM ANNUAL DATA
TRUINT     B      INTEREST RATE CALCULATIØNS

**********MS--MANAGEMENT SCIENCE AND ØPTIMIZATIØN**********

***LINEAR PRØGRAMMING***
LINPRØ     B      LINEAR PRØGRAMMING
LNPRØG     FP     LINEAR PRØGRAMMING
```

```
***INTEGER PROGRAMMING***
INTO1      FP    ZIONTS' MODIFICATION OF BALAS' ZERO-ONE ALGORITHM
INTLP      FP    GOMORY'S PURE AND MIXED INTEGER PROGRAMMING

***NON-LINEAR OPTIMIZATION***
DAVIDON    B     DAVIDON'S UNCONSTRAINED OPTIMIZATION
LOGIC3     FP    UNCONSTRAINED OPTIMIZATION
MAXOPT     FP    UNCONSTRAINED OPTIMIZATION

***NETWORK ANALYSIS***
CPM        FP    CRITICAL PATH METHOD
KILTER     FP    'OUT OF KILTER' ALGORITHM (MINIMUM COST CIRCULATION)
MAXFLOW    FP    MAXIMUM FLOW THRU NETWORK
PERT       B     SIMPLE ANALYSIS OF A PERT NETWORK
SHORTEST   FP    SHORTEST PATH - MIN SPANNING TREE

***FORECASTING***
TCAST      FP    TIME SERIES FORECASTING [TCAST1;TCAST2]
TCAST1     O     OVERLAY MODULE OF TCAST
TCAST2     O     OVERLAY MODULE OF TCAST
SMOOTH     FS    TRIPLE SMOOTHING OF A TIME SERIES

***SIMULATION***
GASPDATA   E     DATA FILE FOR SAMPLE PROGRAM GASPSAMP
GASPIIA    FS    'GASP' SIMULATION SYSTEM
GASPSAMP   FP    SAMPLE PROGRAM FOR GASPIIA [GASPIIA;GASPDATA]

**********EN--ENGINEERING**********

ACNET      FP    FREQUENCY RESPONSE OF A LINEAR CIRCUIT
BEMDES     B     STEEL BEAM SELECTION
GCVSIZ     B     GAS CONTROL VALVE COEFF.
LCVSIC     B     LIQUID CONTROL VALVE COEFF.
LPFILT     B     DESIGN LOW PASS FILTERS
NLNET      FP    GENERAL STEADY-STATE CIRCUIT ANALYSIS
OTTO       B     OTTO CYCLE OF ENGINE
PVT        FP    FINDS MOLAR VOLUME OF A GAS GIVEN TEMPERATURE AND PRES.
SCVSIZ     B     STEAM CONTROL VALVE COEFF.
SECAP      B     STEEL SECTION CAPACITIES

**********GP--GEOMETRIC AND PLOTTING**********

CIRCLE     B     DIVIDES A CIRCLE INTO N EQUAL PARTS
PLOT       FS    PLOTS UP TO 9 CURVES SIMULTANEOUSLY
PLOTTO     B     SIMULTANEOUSLY PLOTS 1 TO 6 FUNCTIONS
POLPLO     FP    PLOTS EONS IN POLAR COORDINATES
SPHERE     B     SOLVES ANY SPHERICAL TRIANGLE
TRIANG     B     SOLVES FOR ALL PARTS OF ANY TRIANGLE
TWOPLO     B     SIMULTANEOUSLY PLOTS 2 FUNCTIONS
XYPLOT     B     PLOTS SINGLE-VALVED FUNCTIONS

**********ED--EDUCATION AND TUTORIAL**********

DRIVES     O     DRIVER FOR EXPER, A COMPUTER ASSISTED INST. LANG.
EXPERN     E     EXPER TUTORIALS IN EXPER (N=1 TO 5) [PREPRS;DRIVES]
PREPRS     O     PREPROCESSOR FOR EXPER, A COMPUTER ASSISTED INST. LANG.

**********DE--DEMONSTRATION**********

BLKJAK     B     THE COMPUTER DEALS BLACKJACK

**********UM--UTILITY AND MISCELLANEOUS**********

CATALOG    E     CATALOG OF SERIES 6000/600 T/S LIBRARY (THIS FILE)
CONVRT     B     CONVERTS MEASUREMENTS FROM ONE SCALE TO ANOTHER
DBLSORT    FS    SORT TWO ARRAYS
SGLSORT    FS    SORT AN ARRAY
TLU1       FS    TABLE SEARCH
TPLSORT    FS    SORT THREE ARRAYS

***END OF CATALOG***
```

4SQRS

This BASIC program writes integers as the sum of squares of four integers.

INSTRUCTIONS

Enter the integer to be partitioned following the "?". The program will continue requesting additional integer until STOP is entered.

SAMPLE PROBLEM

Partition the integers 12, -452, and 39.

SAMPLE SOLUTION

```
*RUN

 4SQRS
 N = A↑2 + B↑2 + C↑2 + D↑2
 N     A      B      C      D
 ?12
       0      2      2      2
 ?-452
       0      0    -14    -16
 ?39
       1      1      1      6
 ?STØP
 READY
 *
```

AMPBX

This FORTRAN subprogram contains two routines to integrate systems of first-order ordinary differential equations by the fourth-order Adams-Moulton method.

INSTRUCTIONS

The two entries in this subprogram are:

CALL AMPB1 (IND,DERIV, TEMP, X, DX, Y, F, N, ICOUNT, NITER, MTST)

CALL AMPB2 (IND,DERIV, TEMP, X, DX, Y, F, N, ICOUNT, NITER, MTST)

where,

- IND is defined as follows:

  IND = 0 indicates the beginning of the integration.

  IND = -1 indicates no adjustment of DX. It is the normal mode and IND will be restored to this value every time the subprogram is called.

  IND = 1 For AMPB1 this indicates that DX be doubled before the next integration step.

  For AMPB2 this indicates that DX be halved before the next integration step.

- DERIV is the name of the derivative routine which must be supplied by the user (an external statement must be used to define DERIV - see sample problem), containing the expressions for the first derivatives of the dependent variables.
- TEMP is the name of a single-dimensioned array containing at least 10* (N+1) elements which must not be used for any other purpose while the integration is being performed.
- X is the value of the independent variable.
- DX is the value of the independent increment.
- Y is the name of the single-dimensioned array containing the dependent variables.
- F is the name of the single-dimensioned array containing the dependent derivatives.
- N is the number of dependent variables.
- ICOUNT is a counter of successive integration steps by the Adams-Moulton method.
- NITER is the number of iterations on the corrector values of the dependent variables. For the normal Adams-Moulton integration, NITER must be set equal to zero.
- MTST is defined as follows:

  MTST = 1 indicates truncation error is treated.

  MTST = 0 indicates truncation error is disregarded.

AMPB1 will compute the derivatives and store the functions and derivatives at each step of the integration. AMPB2 will integrate to the next step. The values of the functions and derivatives at the next step will be stored. However, the integration may be repeated with an adjusted increment. The integration step will be made permanent only by calling AMPB1.

The routine only performs the integration of the differential equations. Provision for output, termination of the integration, and adjustment of the increment must be done by the user. Generally, the output and termination should be done between routine calls to AMPB1 and AMPB2, and adjustment of the increment, if any, should be done after the routine call to AMPB2.

If an adjustment to the increment is desired, it must be done by changing IND. DX may not be adjusted directly by the user.

An approximation of the truncation errors will be stored in TEMP (2) - TEMP (N+1) upon exit from AMPB2 provided ICOUNT ≥ 4.

RESTRICTION

The subprogram RKPBX must be used with this subprogram, as shown in the Sample Problem.

METHOD

For the method used, see reference below.

SAMPLE PROBLEM

Integrate the following system of equations:

$$\frac{dX}{dt} = Y$$

$$\frac{dY}{dt} = -4X$$

$$\frac{dZ}{dt} = 2Z$$

From t = 0 to t = 2 where the interval of integration (dt) is 0.0625 and the initial conditions of X, Y, and Z are:

X = 0.0
Y = 2.0
Z = 1.0

at t = 0. Print t, X, Y, and Z for every integration step.

(In the following program, U(1) is used for X, U(2) is used for Y, and U(3) is used for Z. The derivatives of X, Y, and Z are referred to as F(1), F(2), and F(3), respectively.)

Hildebrand, F. B., Introduction to Numerical Analysis, McGraw-Hill, New York, 1956, Section 6.6.1.

SAMPLE SOLUTION

```
010       EXTERNAL DERIV
020       COMMON U(3),F(3)
030       DIMENSION TEMP(40)
040       T=0.0; DT=0.0625; N=3; U(1)=0.0; U(2)=2.0; U(3)=1.0
050       TF=2.0; IND=0
060       PRINT 1
070    1  FORMAT(/8X,1HT,13X,1HX,13X,1HY,13X,1HZ)
080       NITER=0; MTST=0
090    10 CALL AMPB1(IND,DERIV,TEMP,T,DT,U,F,N,ICOUNT,NITER,MTST)
100       PRINT 15, T,U
110    15 FORMAT(1H 4E14.6)
120       IF (T-TF) 20,100,100
130    20 CALL AMPB2(IND,DERIV,TEMP,T,DT,U,F,N,ICOUNT,NITER,MTST)
140       GO TO 10
150   100 STOP
155       END
160C   DERIVATIVE EVALUATION SUBROUTINE
170       SUBROUTINE DERIV
180       COMMON U(3),F(3)
190       F(1)=U(2); F(2)=-4.0*U(1); F(3)=2.0*U(3)
200       RETURN
210       END
```

READY

```
*RUN *;AMPBX;RKPBX
```

T	X	Y	Z
0.	0.	0.200000E+01	0.100000E+01
0.625000E-01	0.124674E+00	0.198440E+01	0.113315E+01
0.125000E+00	0.247403E+00	0.193782E+01	0.128402E+01
0.187500E+00	0.366272E+00	0.186102E+01	0.145499E+01
0.250000E+00	0.479426E+00	0.175517E+01	0.164872E+01
0.312500E+00	0.585098E+00	0.162193E+01	0.186825E+01
0.375000E+00	0.681641E+00	0.146338E+01	0.211700E+01
0.437500E+00	0.767546E+00	0.128199E+01	0.239888E+01
0.500000E+00	0.841475E+00	0.108060E+01	0.271828E+01
0.562500E+00	0.902272E+00	0.862350E+00	0.308022E+01
0.625000E+00	0.948990E+00	0.630640E+00	0.349035E+01
0.687500E+00	0.980899E+00	0.389088E+00	0.395508E+01
0.750000E+00	0.997501E+00	0.141465E+00	0.448170E+01
0.812500E+00	0.998537E+00	-0.108367E+00	0.507843E+01
0.875000E+00	0.983991E+00	-0.356508E+00	0.575462E+01
0.937500E+00	0.954090E+00	-0.599085E+00	0.652084E+01
0.100000E+01	0.909301E+00	-0.832315E+00	0.738908E+01
0.106250E+01	0.850322E+00	-0.105256E+01	0.837293E+01
0.112500E+01	0.778074E+00	-0.125637E+01	0.948777E+01
0.118750E+01	0.693684E+00	-0.144058E+01	0.107511E+02
0.125000E+01	0.598470E+00	-0.160232E+01	0.121825E+02
0.131250E+01	0.493916E+00	-0.173904E+01	0.138046E+02
0.137500E+01	0.381654E+00	-0.184863E+01	0.156427E+02
0.143750E+01	0.263437E+00	-0.192938E+01	0.177255E+02
0.150000E+01	0.141108E+00	-0.198001E+01	0.200856E+02

```
 0.156250E+01  0.165779E-01 -0.199975E+01  0.227600E+02
 0.162500E+01 -0.108211E+00 -0.198828E+01  0.257905E+02
 0.168750E+01 -0.231312E+00 -0.194579E+01  0.292245E+02
 0.175000E+01 -0.350804E+00 -0.187293E+01  0.331157E+02
 0.181250E+01 -0.464821E+00 -0.177084E+01  0.375250E+02
 0.187500E+01 -0.571585E+00 -0.164113E+01  0.425214E+02
 0.193750E+01 -0.669430E+00 -0.148580E+01  0.481831E+02
 0.200000E+01 -0.756828E+00 -0.130728E+01  0.545986E+02

PROGRAM STOP AT 150
*
```

ARCTAN

This FORTRAN subprogram determines an angle (in radians) on the interval ($-\pi$, π) whose tangent is Y/X.

INSTRUCTIONS

The calling sequence for the entry ARCTAN is:

CALL ARCTAN(X, Y, ANGLE, ERROR).

where,

- X, Y and ANGLE are defined above.
- ERROR is an error flag; when the routine returns to the calling program, the error will contain 1 if X = Y = 0. Otherwise, it will contain 0.

SAMPLE PROBLEM

Find the angle whose tangent is 1/1, -1/-1, and 2/-1.

SAMPLE SOLUTION

```
*LIST

10 PRINT 1
15 1 FORMAT("0          X                   Y            ARCTAN(Y/X)")
20 Y=1;X=1
25 CALL ARCTAN(X,Y,A,E)
30 IF(E)2,3,2
35 2 PRINT:"E"
40 STOP
45 3 PRINT 4,X,Y,A
50 4 FORMAT(1P3E16.7)
55 X=-1;Y=-1
60 CALL ARCTAN(X,Y,A,E)
65 IF(E)2,5,2
70 5 PRINT 4,X,Y,A
75 Y=2.
80 CALL ARCTAN(X,Y,A,E)
85 IF(E)2,6,2
90 6 PRINT 4,X,Y,A
95 STOP;END

READY

*RUN *;ARCTAN

          X                   Y            ARCTAN(Y/X)
  1.0000000E+00   1.0000000E+00   7.8539816E-01
 -1.0000000E+00  -1.0000000E+00  -2.3561945E+00
 -1.0000000E+00   2.0000000E+00   2.0344439E+00

PROGRAM STOP AT 95
*
```

BESL

This FORTRAN subroutine computes the Bessel function, J, of the first kind, for a real order and real argument.

INSTRUCTIONS

The calling sequence for the entry BESL is:

CALL BESL (IND, ORD, X, XJ)

where,

- IND determines the number of Bessel functions returned, as follows:

  IND = 1, one answer is returned.

  IND = 2, a set of Bessel functions is returned.

- ORD is the order of the Bessel function and can be any real number.
- X, the argument, is any nonnegative real number.
- XJ is a single dimension array into which the results are stored.

RESTRICTION

The subprogram GAMF must be used with BESL, as shown in the Sample Solution.

METHOD

If IND = 2, the values returned correspond to Bessel functions of orders V to $\pm$N + V in steps of 1, evaluated at X, where ORD = $\pm$N + V, N is an integer and V is a real number not less than zero but less than 1. N and V are determined by the program.

XJ must be a dimension variable, i.e., XJ(M) where M is determined as follows:

if IND = 1, M = 1

If IND = 2, M must be at least $\lfloor |\text{ORD}| \rfloor + 2$

It is recommended that IND be interrogated after a call to BESL as a check on accumulator overflow, because if this occurs IND is set to - IND and a return is made to the main program.

A backward recurrence method is used[1].

[1] Bessel Functions of Integer and Fractional Order, Handbook of Mathematical Functions, National Bureau of Standards.

SAMPLE PROBLEM

Find:

$J_0(5)$

$J_1(3)$

$J_X(5)$, X = 0, 1, 2, 3

SAMPLE SOLUTION

```
10      DIMENSIØN XJ(10)
15      PRINT 2
20    2 FØRMAT("        ØRDER          ARGUMENT          FUNCTIØN")
25      IND=1;ØRD=0.;X=5.
30      CALL BESL(IND,ØRD,X,XJ)
40      PRINT 1,ØRD,X,XJ(1)
50      ØRD=1.;X=3.
60      CALL BESL(IND,ØRD,X,XJ)
70      PRINT 1, ØRD,X,XJ(1)
75    1 FØRMAT(3E16.8)
80      IND=2;ØRD=3;X=5.
85      CALL BESL(IND,ØRD,X,XJ)
90      DØ 3 I=1,4
92      P=I-1
93    3 PRINT 1,P,X,XJ(I)
95      STØP;END

READY

*RUN *;BESL;GAMF
      ØRDER          ARGUMENT          FUNCTIØN
 0.                0.50000000E+01  -0.17759678E+00
 0.10000000E+01    0.30000000E+01   0.33905895E+00
 0.                0.50000000E+01  -0.17759678E+00
 0.10000000E+01    0.50000000E+01  -0.32757915E+00
 0.20000000E+01    0.50000000E+01   0.46565115E-01
 0.30000000E+01    0.50000000E+01   0.36483124E+00

PRØGRAM STØP AT 95
*
```

BICOF

This FORTRAN subroutine generates a set of binomial coefficients.

INSTRUCTIONS

The calling sequence for the entry BICOF is:

```
CALL BICOF(I, N, COF)
```

where,

- I and N are indices indicating the range of coefficients.
- COF is the name of the coefficient array.

If I is less than N, COF will contain the I through N coefficients. If I is greater than N, COF will contain the N through I coefficients. If I is equal to N, COF will contain the Nth coefficient.

SAMPLE PROBLEM

Find the coefficients of $(A + B)^3$ and $(A + B)^7$. Find the last 3 coefficients of $(A + B)^5$.

SAMPLE SOLUTION

```
10      DIMENSION C0F(10)
20      CALL BIC0F(0,3,C0F)
25      PRINT 1,(C0F(I),I=1,4)
30      CALL BIC0F(0,7,C0F)
35      PRINT 1,(C0F(I),I=1,8)
40      CALL BIC0F(3,5,C0F)
50      PRINT 1,(C0F(I),I=1,3)
60    1 F0RMAT(/8F6.2)
70      ST0P;END

READY

*RUN *;BIC0F

 1.00   3.00   3.00   1.00

 1.00   7.00  21.00  35.00  35.00  21.00   7.00   1.00

10.00   5.00   1.00

PR0GRAM ST0P AT 70
*
```

BROWN

This FORTRAN subroutine solves a system of n simultaneous nonlinear equations in n unknowns using Brown's algorithm (Comm. ACM vol. 10 page 728). The algorithm is a modification of Newton's method requiring no derivative evaluations.

INSTRUCTIONS

The subroutine is called as below

CALL BROWN (N, MAXIT, EPS, ISING, X, FUNCT)

where on input N is the number of equations, MAXIT is an upper bound on the number of iterations, EPS is a small number used to test for convergence, and X is the vector of initial guesses to the solution. FUNCT is the external subroutine supplied by the user. BROWN calls FUNCT as below

CALL FUNCT (X, FK, K)

When called FUNCT should evaluate the K^{th} function at X (X is a vector) and return the value in FK.

On exit from BROWN, the vector X is the solution of the system (or its best approximation) and MAXIT is the actual number of iterations performed. ISING = 0 if a Jacobian related matrix was singular (the routine was "blowing-up") or ISING = 1 if no such difficulty was found.

Dimension statements limit N to be less than or equal to 20.

SAMPLE PROBLEM

Solve the system

$$\left(1 - \frac{1}{4\Pi}\right)\left(e^{2X} - e\right) + \frac{e}{\Pi}y - 2eX = 0$$

$$1/2 \sin(xy) - \frac{y}{4\Pi} - \frac{X}{2} = 0$$

One solution of which is (.5, Π).

SAMPLE SOLUTION

```
*LIST

0010 DIMENSIØN X(20)
0020 EXTERNAL FUNCT
0030 X(1)=.7;X(2)=2.8
0040 MAXIT=50
0050 CALL BRØWN(2,MAXIT,1E-4,ISING,X,FUNCT)
0060 PRINT:"ISING= ",ISING," MAXIT= ",MAXIT
0070 PRINT:"SØLUTIØN",X(1),X(2)
0080 STØP;END
0090 SUBRØUTINE FUNCT(X,FK,K)
0100 DIMENSIØN X(20)
0110 GØ TØ (1,2),K
0120 1 FK=2.71828183*(.920422528*(EXP(2*X(1)-1)-1)+X(2)/3.14159265
0121& -2.*X(1))
0130 RETURN
0140 2 FK=.5*SIN(X(1)*X(2))-X(2)/12.5663706-X(1)/2
0150 RETURN
0160 END

READY

*RUN *;BRØWN
ISING=             1 MAXIT=             7
SØLUTIØN    4.9999999E-01    3.1415926E+00

PRØGRAM STØP AT 80
*
```

CLCINT

This FORTRAN function computes the integral

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

by the Trapezoidal Rule or Simpson's Rule.

INSTRUCTIONS

The calling sequence for the entry CLCINT is:

```
Y = CLCINT(IND,DX,FX,TEMP)
```

where,

- Y is the value of the integral.
- IND is defined as: IND = 0 Trapezoidal Rule
  IND = 1 Simpson's Rule
- DX is defined as: DX = 0 When X = A
  DX = Integration Increment When X ≠ A.
- FX is the integrand.
- TEMP is an array of dimension 5, which must not be used for any other purpose while the integration is being performed.

METHOD

The first interval is always computed by Trapezoidal Rule.

If IND = 0, subsequent intervals are computed by Trapezoidal Rule. If IND = 1, subsequent intervals are computed by Simpson's Rule (if the current and previous values of DX are equal; otherwise, they are computed by Trapezoidal Rule). Assuming a constant DX, the net effect of this procedure is: If N is odd, the integral consists of the Trapezoidal Rule integration over the first interval and Simpson's Rule integration over the remaining N-1 intervals. If N is even, the integral consists of Simpson's Rule integration over the N intervals[1].

SAMPLE PROBLEM

Evaluate by Simpson's Rule the integral of the function SIN(X), in the interval zero to 2π radians using an increment DX = .01.

[1] Hamming, R.W., Numerical Method for Scientists and Engineers, McGraw-Hill, New York, 1962, Section 13.2.

SAMPLE SOLUTION

```
10      DIMENSIØN TEMP(5)
20      X=0.0;DX=0.0
30      Y=CLCINT(1,DX,SIN(X),TEMP)
40      DØ 10 J=1,628
50      A=J
60      X=A*.01
70   10 Y=CLCINT(1,.01,SIN(X),TEMP)
80      PRINT 15, Y
90   15 FØRMAT("0       VALUE ØF THE INTEGRAL =",1PE20.7)
100      STØP;END

READY

*RUN *;CLCINT

       VALUE ØF THE INTEGRAL =        2.0979569E-06

PRØGRAM STØP AT 100
*
```

CLPLY

This FORTRAN function evaluates a polynomial defined as the sum of powers of a single real variable.

INSTRUCTIONS

The calling sequence for the entry CLPLY is:

```
Y=CLPLY(X,A,N)
```

where,

- Y is the value of the polynomial.
- X is the value of the independent variable.
- A is the name of the coefficient array (stored constant term first).
- N is the degree of the polynomial.

METHOD

The standard nesting process is used.

SAMPLE PROBLEM

Evaluate the following polynomial at X=1.0:

$$3X^5 - X^2 + 2X - 5$$

SAMPLE SOLUTION

```
10      DIMENSIØN A(6)
20      A(1)=-5.0
30      A(2)=2.0
35      A(3)=-1.0
40      A(4)=0.0;A(5)=0.0
45      A(6)=3.0
50      Y=CLPLY(1.,A,5)
55      PRINT 10,Y
60   10 FØRMAT(/26H VALUE ØF THE PØLYNØMIAL =,1PE20.7)
65      STØP;END

READY

*RUN *;CLPLY

VALUE ØF THE PØLYNØMIAL =       -1.0000000E+00

PRØGRAM STØP AT 65
*
```

COMP1

This FORTRAN function evaluates the real functions: hyperbolic sine, hyperbolic cosine hyperbolic tangent, arcsine, and arccosine.

INSTRUCTIONS

The calling sequence for the entry COMP1 is:

Y = COMP1(IND, A)

where,

- Y is the value of the function.

  IND = 1 real hyperbolic sine
  IND = 2 real hyperbolic cosine
  IND = 3 real hyperbolic tangent
  IND = 4 real arcsine
  IND = 5 real arccosine

- A is the input argument. For IND = 1, 2, 3, A must be in radians. For IND = 4 and 5. Y will be returned as radians.

RESTRICTIONS

Hyperbolic sine, hyperbolic cosine: |A| must be less than 88.

Hyperbolic tangent: none.

Arcsine, arccosine: |A| must be less than or equal to 1.0.

METHOD

Evaluation of power series and use of exponential and square root functions, depending on the range of the argument and the function desired.

SAMPLE PROBLEM

Find the hyperbolic sine, cosine, and tangent of the first five non-negative integers.

Find the angle whose sine is 0.885235471.

Find the angle whose cosine is 0.574338891.

SAMPLE SOLUTION

```
*LIST

10 PRINT 1
15 1 FØRMAT("0",7X,"X",13X,"SINH(X)",9X,"CØSH(X)",9X,"TANH(X)"//)
20 DØ 2 I=1,5
25 A=I
30 PRINT 3,A,CØMP1(1,A),CØMP1(2,A),CØMP1(3,A)
35 2 CØNTINUE
40 3 FØRMAT(1P4E16.7)
45 A=.885235471
50 PRINT 4,A,CØMP1(4,A)
55 4 FØRMAT("0THE ANGLE WHØSE SINE IS ",F10.6," IS ",F10.6)
60 A=.574338891
65 PRINT 5,A,CØMP1(5,A)
70 5 FØRMAT("0THE ANGLE WHØSE CØSINE IS ",F10.6," IS ",F10.6)
75 STØP;END

READY

*RUN *;CØMP1

      X              SINH(X)          CØSH(X)          TANH(X)

  1.0000000E+00   1.1752012E+00   1.5430806E+00   7.6159415E-01
  2.0000000E+00   3.6268604E+00   3.7621957E+00   9.6402758E-01
  3.0000000E+00   1.0017875E+01   1.0067662E+01   9.9505475E-01
  4.0000000E+00   2.7289917E+01   2.7308233E+01   9.9932930E-01
  5.0000000E+00   7.4203210E+01   7.4209948E+01   9.9990920E-01

THE ANGLE WHØSE SINE IS   0.885235 IS   1.087000

THE ANGLE WHØSE CØSINE IS   0.574339 IS   0.959000

PRØGRAM STØP AT 75
*
```

COMP2

This FORTRAN subroutine evaluates a complex product or quotient.

INSTRUCTIONS

The calling sequence for this subprogram is:

CALL COMP2(IND, AR, AI, BR, BI, CR, CI)

Complex Multiplication

IND = 1

AR is the real part of the multiplier
AI is the imaginary part of the multiplier
BR is the real part of the multiplicand
BI is the imaginary part of the multiplicand
CR is the real part of the product
CI is the imaginary part of the product

Complex Division

IND = 2

AR is the real part of the dividend
AI is the imaginary part of the dividend
BR is the real part of the divisor
BI is the imaginary part of the divisor
CR is the real part of the quotient
CI is the imaginary part of the quotient

RESTRICTION

If BR=BI=0, the largest number possible is returned for the quotient CR and CI(IND = 2). This number is approximately 10^{38}

METHOD

For complex multiplication (IND = 1), the method is:

CR=AR*BR-AI*BI
CI =AR*BI +AI*BR

For complex division (IND = 2), the method is:

CR=(X2*X4+X1)/X3
CI =(X2-X1*X4)/X3

where,

ABSF(BR)< ABSF(BI)	X1= AI/BI X2=-AR/BI X4=-BR/BI	X3=1. +X4*X4
ABSF(BR)≥ ABSF(BI)	X1= AR/BR X2= AI/BR X4= BI/BR	X3=1. +X4*X4

COMP2

SAMPLE PROBLEM

Multiply and divide the two complex numbers 2+2i and 1-4i

SAMPLE SOLUTION

```
10      IND=1
20      AR=2.0;AI=2.0
30      BR=1.0
40      BI=-4.0
50   20 CALL COMP2(IND,AR,AI,BR,BI,CR,CI)
60      PRINT 30, IND,CR,CI
70   30 FORMAT("0IND=",I5,10X,"CR=",1PE14.6,5X,"CI=",1PE14.6)
80      IF (IND-2) 40,50,40
90   40 IND=2
100      GO TO 20
110  50 STOP;END

READY

*RUN *;COMP2

IND=     1          CR=  1.000000E+01      CI= -6.000000E+00

IND=     2          CR= -3.529412E-01      CI=  5.882353E-01

PROGRAM STOP AT 110
*
```

RESTRICTION

If BR=BI=0, the largest number possible is returned for the quotient CR and CI(IND = 2).
This number is approximately 10^{38}

METHOD

For complex multiplication (IND = 1), the method is:

CR=AR*BR-AI*BI
CI =AR*BI +AI*BR

For complex division (IND = 2), the method is:

CR=(X2*X4+X1)\X3
CI =(X2-X1*X4)\X3

where,

ABSF(BR)< ABSF(BI) X1= AI\BI
X2=-AR\BI
X4=-BR\BI X3=1.+X4*X4

ABSF(BR)≥ ABSF(BI) X1= AR\BR
X2= AI\BR
X4= BI\BR X3=1.+X4*X4

COMP3

This FORTRAN subroutine evaluates the complex functions:

exponential
square root
sine {radian / argument}
cosine {radian / argument}
modulus

logarithm (base e)
hyperbolic sine {radian / argument}
hyperbolic cosine {radian / argument}
natural logarithm of the gamma function

INSTRUCTIONS

The calling sequence for this subroutine is:

CALL COMP3(IND, AR, AI, CR, CI)

where,

- IND = 1 Complex exponential
- IND = 2 Complex square root
- IND = 3 Complex sine {radian / argument}
- IND = 4 Complex cosine {radian / argument}
- IND = 5 Modulus
- IND = 6 Complex logarithm
- IND = 7 Complex hyperbolic sine {radian / argument}
- IND = 8 Complex hyperbolic cosine {radian / argument}
- IND = 9 Natural log of the complex gamma function

AR and AI are the real and imaginary parts of the input arguments and CR and CI are the real and imaginary parts of the answer, except for the modulus routine where there is only one answer. The answer is returned in CR; however, CI must be included in the calling sequence.

RESTRICTION

The subprogram COMP2 must be used with this subroutine. For an example of this use, see the Sample Problem.

METHOD

The following procedures are used in COMP3.

Complex Exponentiation

When IND = 1, the method is:

```
CR=EXPF(AR)*COSF(AI)
CI =EXPF(AR)*SINF(AI)
```

where,

- The magnitude of AR must be less than or equal to 88; otherwise, if the argument is negative, both results are set to zero. If the argument is positive, the answer returned is the largest number possible, approximately 10^{38}
- The magnitude of AI must be less than or equal to 2^{27}; otherwise, the answers, CR and CI, are set to 0.

Complex Square Root

When IND = 2, the method is:

```
If AR ≥ 0.    CR=X
              CI =AI/(2.0*X)

If AR < 0.    CR=ABSF(AI/(2.0*X))
              CI =SIGNF(X,AI)
```

where,

- X=SQRTF((ABSF(AR)+CABS(AR,AI))/2.0)

and

- CABS indicates the modulus function.

Of the two roots, the root in the right-hand plane is returned as the answer. For the special case of a real, negative input, the returned root lies on the positive imaginary axis.

Complex Sine

When IND = 3, the method is:

```
CR=SINF (AR)*COSH(AI)
CI =COSF(AR)*SINH(AI)
```

where,

- COSH and SINH are the hyperbolic cosine and sine functions.
- The magnitude of AR must be less than or equal to 2^{27}; otherwise, the answers, CR and CI, are set to 0.
- The magnitude of AI must be less than or equal to 88; otherwise, the answers returned are the largest numbers possible, approximately 10^{38}.

Complex Cosine

When IND = 4, the method is:

```
CR=COSF(AR)*COSH(AI)
CI =SINF (AR)* SINH(AI)
```

where,

- COSH and SINH are the hyperbolic cosine and sine functions.
- ABSF(AR) and ABSF(AI) are the same as for IND = 3.

Modulus

When IND = 5, the method is:

```
CR=ABSF(L)*SQRTF(1.0+(S/L)**2)

If ABSF(AR) ≥ ABSF(AI)    L=AR
                          S=AI

If ABSF(AR) < ABSF(AI)    L=AI
                          S=AR
```

The value is returned in CR; CI is not used.

Complex Logarithm

When IND = 6, the method is:

```
CR=LOGF(C)
CI=ATANF(AI/AR
```

where, C is the modulus of the complex argument. The value of CI is chosen such that

$$\pi - < CI \leq \pi$$

In the special case, where AR=AI=0, the answers returned are approximately -10^{38}.

Complex Hyperbolic Sine

When IND = 7, the method is:

```
CR=(EXPF(AR)*COSF(AI)-EXPF(-AR)*COSF(-AI))/2.
CI =(EXPF(AR)*SINF(AI)-EXPF(-AR)*SINF(-AI))/2.
```

where,

- The magnitude of AR must be less than or equal to 88; otherwise, the answers returned are the largest possible, approximately 10^{38}.
- The magnitude of AI must be less than or equal to 2^{27}; otherwise, the answers, CR and CI, are set to 0.

Complex Hyperbolic Cosine

When IND = 8, the method is:

```
CR=(EXPF(AR)*COSF(AI)+EXPF(-AR)*COSF(-AI))/2.
CI =(EXPF(AR)*SINF(AI)+EXPF(-AR)*SINF(-AI))/2.
```

Requirements for the magnitudes of AR and AI are the same as when IND = 7.

Natural Log of the Complex Gamma Function

When IND = 9, the method is as in the reference below. The requirements are:

- If AR=AI=0, CR and CI are set to 0.
- If AR is a negative integer and AI=0, CR and CI are set to 0.

SAMPLE PROBLEM

Find the complex exponential, complex square root, complex sine, complex cosine, modulus, complex logarithm, complex hyperbolic sine, complex hyperbolic cosine, and natural log of the complex gamma function for the complex number, 1+1i.

Lanczos, C., "A Precision Approximation of the Gamma Function," Journal of SIAM, Numerical Analysis, Series B, Volume 1, 1964.

SAMPLE SOLUTION

```
10      IND=1
20      AR=1.0;AI=1.0
30   20 IF (IND-10)25,80,25
40   25 CALL COMP3(IND,AR,AI,CR,CI)
50      IF (IND-5) 40,60,40
60   40 PRINT 45,IND,CR,CI
70   45 FORMAT("0IND=",I5,10X,"CR=",1PE14.6,5X,"CI=",1PE14.6)
80      GO TO 50
90   60 PRINT 70, IND,CR
100   70 FORMAT("0IND=",I5,10X,"CR=",1PE14.6)
110   50 IND=IND+1
115      GO TO 20
120   80 STOP;END

READY

*RUN *;COMP3;COMP2

IND=     1          CR=  1.468694E+00     CI=  2.287355E+00

IND=     2          CR=  1.098684E+00     CI=  4.550899E-01

IND=     3          CR=  1.298458E+00     CI=  6.349639E-01

IND=     4          CR=  8.337300E-01     CI= -9.888977E-01

IND=     5          CR=  1.414214E+00

IND=     6          CR=  3.465736E-01     CI=  7.853982E-01

IND=     7          CR=  6.349639E-01     CI=  1.298458E+00

IND=     8          CR=  8.337300E-01     CI=  9.888977E-01

IND=     9          CR= -6.509233E-01     CI= -3.016404E-01

PROGRAM STOP AT 120
*
```

DETE

This FORTRAN function evaluates the determinant of a matrix of real elements.

INSTRUCTIONS

The calling sequence for the entry DETE is:

X = DETE (A, N, IDIM)

where,

- X is the value of the determinant
- A is the name of the array of matrix elements. This matrix is altered during the course of the evaluation.
- N is the order of the matrix.
- IDIM is the first dimension of the array A, i.e., A(IDIM, N).

METHOD

The solution is obtained by the Triangular method[1].

SAMPLE PROBLEM

Evaluate the determinant of the following matrix:

2.	0.	1.	-3.
4.	1.	-2.	0.
-3.	4.	2.	5.
0.	1.	0.	1.

[1] Scarborough, J. B., Numerical Mathematical Analysis, Sixth Edition, The Johns Hopkins Press, Baltimore, Maryland, 1962.

SAMPLE SOLUTION

```
010       DIMENSIØN A(25,4)
020       A(1,1)=2.0;A(3,3)=2.0
030       A(2,1)=4.0;A(3,2)=4.0
040       A(3,1)=-3.0;A(1,4)=-3.0
050       A(4,1)=0.0;A(1,2)=0.0;A(4,3)=0.0;A(2,4)=0.0
060       A(2,2)=1.0;A(4,2)=1.0;A(1,3)=0.0;A(4,4)=0.0
070       A(2,3)=-2.0
080       A(3,4)=5.0
090       X=DETE(A,4,25)
100       PRINT 10,X
110    10 FØRMAT(/27H VALUE ØF THE DETERMINANT =,1PE20.7)
120       STØP
130       END

READY

*RUN *;DETE

VALUE ØF THE DETERMINANT =       -2.6000000E+01

PRØGRAM STØP AT 120
*
```

DOMEIG

This FORTRAN program calculates either the dominant or subdominant eigenvalue and the eigenvector of a real, square matrix.

METHOD

The method used is the Power Method of Misus. If the matrix is nonsingular, has a dominant eigenvalue, and has N linearly-independent eigenvectors, the dominance of the eigenvalue can be used to extract it and its eigenvector. If the eigenvector has multiplicity greater than one, the eigenvector found is just one of the general family corresponding to the eigenvalue. To find the subdominant eigenvalue the matrix is inverted and the inverse matrix will have replaced the original upon completion.

INSTRUCTIONS

To use this program enter data as requested.

DOMEIG can also be used as a subroutine. This is done by deleting lines 1 through 99. The calling sequence for the entry DOMEIG is:

```
CALL DOMEIG (A, VI, N, EIGEN, DM, ITMAX)
```

where,

- A is the name of a two dimensional array containing the N by N matrix whose dominant or subdominant eigenvalue is to be found.
- VI is the name of a one dimensional array containing an initial guess of the eigenvector and containing the eigenvector upon completion.
- N is the dimension of the matrix.
- EIGEN is the real variable in which the eigenvalue will be stored upon completion.
- DM is a variable specified by the user as a zero if the subdominant eigenvalue is to be found. Otherwise, the dominant eigenvalue will be found. It indicates the result of the search for the eigenvalue by returning with one of the following values:

  0 - subdominant eigenvalue is zero because matrix was singular.

  1 - The dominant eigenvalue and eigenvector were found.

  2 - ITMAX was exceeded and the eigenvalue was not found.

  3 - The initial guess was an eigenvector and therefore, the eigenvalue found may not be dominant or subdominant.

  4 - The subdominant eigenvector and eigenvalue were found.

- ITMAX is the maximum number of iterations to be used in looking for the eigenvalue. It is typically in the order of 50 to 100. If the subroutine is called with ITMAX zero, then ITMAX is set to 100 by the subroutine.

NOTE:

If the subdominant eigenvalue is to be found, the original matrix is destroyed and the inverse created in its place.

SAMPLE PROBLEM

Find the subdominant eigenvector of the matrix

$$\begin{bmatrix} 10 & 9 & 7 & 5 \\ 9 & 10 & 8 & 6 \\ 7 & 8 & 10 & 7 \\ 7 & 5 & 6 & 7 \end{bmatrix}$$

SAMPLE SOLUTION

```
 *RUN DØMEIG
ENTER THE ØRDER ØF MATRIX.
= 4
ENTER THE MATRIX CØLUMN BY CØLUMN.  SEPARATE EACH ITH
A CØMMA, BUT DØ NØT PUT A CØMMA AT THE END.
= 10,9,7,5
= 9,10,8,6
= 7,8,10,7
= 7,5,6,7
ENTER 1 FØR DØMINANT EIGENVALUE.
ENTER 0 FØR SUBDØMINANT EIGENVALUE.
= 0
ENTER THE NUMBER ØF ITERATIØNS TØ BE USED IN LØØKING
FØR THE EIGENVALUE.  ENTER A 0 IF YØU WØULD LIKE
THE SUBRØUTINE TØ SET IT FØR YØU.
= 50
ENTER EIGENVECTØR GUESS.
= 1,1,1,1
SUBDØMINANT EIGENVALUE AND EIGENVECTØR
    1.7000001E+01
    4.7946327E-02    4.7946329E-02    8.1508760E-01   -5.7535598E-01

PRØGRAM STØP AT 99
*
```

DVALG

This FORTRAN subroutine finds the quotient of two polynomials and the resulting remainder, if any.

INSTRUCTIONS

The calling sequence for the entry DVALG is:

```
CALL DVALG(A,NA,B,NB,Q,R,NR,TEST)
```

where,

- A is the name of the dividend coefficient array.
- NA is the degree of the dividend polynomial.
- B is the name of the divisor coefficient array.
- NB is the degree of the divisor polynomial.
- Q is the name of the quotient coefficient array.
- R is the name of the remainder coefficient array.
- NR is the degree of the remainder polynomial.
- TEST is the criterion for determining zero coefficients of the remainder polynomial as shown in the Sample Problem.

All polynomial coefficients are stored constant term first.

RESTRICTION

NB must not exceed NA, which in turn must not exceed 25.

SAMPLE PROBLEM

Divide the polynomial $3X^4 + 2X^2 + X + 4$ by the polynomial $X^2 - 1$ and print the coefficients of both the quotient and remainder, considering any remainder coefficient with a magnitude less than 1.0×10^{-7} to be zero.

SAMPLE SOLUTION

```
*10 DIMENSION A(5),B(3),Q(3),R(3)
*20 A(1)=4.0
*30 A(2)=1.0
*40 A(3)=2.0
*50 A(4)=0.0
*60 A(5)=3.0
*70 B(1)=-1.0
*80 B(2)=0.0
*90 B(3)=1.0
*100 TEST=1.0E-7
*110 CALL DVALG(A,4,B,2,Q,R,NR,TEST)
*120 PRINT 10
*130 10 FORMAT(/22H QUOTIENT COEFFICIENTS)
*140 DO 15 I=1,3
*150 M=I-1
*160 15 PRINT 20, M,Q(I)
*170 20 FORMAT(/19H COEFFICIENT OF X**,I2,1PE20.7)
*180 PRINT 25
*190 25 FORMAT(/23H REMAINDER COEFFICIENTS)
*200 DO 30 I=1,3
*210 30 PRINT 20,I-1,R(I)
*220 STOP
*230 END
*RUN *;DVALG

QUOTIENT COEFFICIENTS

COEFFICIENT OF X** 0        5.0000000E+00

COEFFICIENT OF X** 1        0.

COEFFICIENT OF X** 2        3.0000000E+00

REMAINDER COEFFICIENTS

COEFFICIENT OF X** 0        9.0000000E+00

COEFFICIENT OF X** 1        1.0000000E+00

COEFFICIENT OF X** 2        0.

PROGRAM STOP AT 220
*
```

EIG1

This FORTRAN program finds the eigenvalues and eigenvectors of a real symmetric matrix by the JACOBI-CORBATO method.

INSTRUCTIONS

The calling sequence for entry EIG1 is:

CALL EIG1 (A, B, N, EPS, TEMP1, TEMP2, IDIMA, IDIMB)

where,

- A is the name of the array containing the elements of the matrix. There are two options:

  IDIMA . EQ. 1. This signifies that A is a single dimensioned variable in which the elements of the upper triangular portion of the matrix are stored contiguously and row-wise.

  IDIMA . GT. 1. This signifies that A is a double dimensioned variable in which the elements of the upper triangular portion of the matrix are stored; i. e., A(IDIMA, J).

- B is the name of the array in which the eigenvectors are stored column-wise.
- N is the order of the matrix. (Must be greater than 1)
- EPS is the convergence criterion The sum of the squares of the off diagonal elements will be less than EPS. If EPS=O., 1.OE-12 will be used as EPS.
- TEMP1 and TEMP2 are the names of two arrays containing at least N cells each that are used for internal storage.
- IDIMA - see A.
- IDIMB is the first dimension of the B array; i. e., B (IDIMB, K).

The eigenvalues are stored in the first N cells of A, where IDIMA equals 1 or in the principal diagonal elements of A where IDIMA is greater than 1.

SAMPLE PROBLEM

Find the eigenvalues and vectors of the real symmetric matrix

$$\begin{bmatrix} 7. & -2. & 0. & 0. \\ -2. & 7. & -2. & -1. \\ 0. & -2. & 7. & 0. \\ 0. & -1. & 0. & 7. \end{bmatrix}$$

use the option where IDIMA = 1, i. e., store the upper triangular elements row-wise in a single dimensioned variable. Iterate until the sum of the squares of the off diagonal elements is less than 1.OE-12.

SAMPLE SOLUTION

```
10      DIMENSIØN A(10),B(4,4),TEMP1(4),TEMP2(4)
20      DATA A/7.,-2.,0.,0.,7.,-2.,-1.,7.,0.,7./
30      CALL EIG1(A,B,4,1.0E-12,TEMP1,TEMP2,1,4)
40      DØ 10 I=1,4
50      PRINT 20,I,A(I)
60   10 PRINT 30,I,(B(J,I),J=1,4)
70   20 FØRMAT(11HOEIGENVALUE,I2,1PE20.7)
80   30 FØRMAT(7H VECTØR,I2,1P4E15.6)
90      STØP;END

READY

*RUN *;EIG1

EIGENVALUE 1         6.9999995E+00
VECTØR 1    7.130160E-01  -1.681058E-08  -7.007160E-01  -2.460006E-02

EIGENVALUE 2         9.9999985E+00
VECTØR 2   -4.714045E-01   7.071067E-01  -4.714046E-01  -2.357023E-01

EIGENVALUE 3         3.9999993E+00
VECTØR 3    4.714045E-01   7.071067E-01   4.714045E-01   2.357022E-01

EIGENVALUE 4         6.9999992E+00
VECTØR 4   -2.171720E-01   5.154200E-09  -2.540721E-01   9.424880E-01

PRØGRAM STØP AT 90
*
```

EIGSR

This FORTRAN program computes the Eigenvalues and Eigenvectors of a real symmetric matrix.

INSTRUCTIONS

The instructions for providing input to this program are generated by the program itself (see the output of the sample solution). If the user types a matrix element incorrectly, the program permits the element to be corrected. The program also generates the instructions for correcting input typing errors (see sample problem).

Whenever the Eigenvalues and Eigenvectors of a given case have been computed the user is then given the opportunity to insert another real symmetric matrix. This program loop will continue until the user types 0 (zero) in response to the question "ORDER?".

RESTRICTIONS

The maximum-size, real-symmetric matrix that can be accepted by the program is a 25 x 25.

NOTE:

This program calls the subroutine EIG1.

SAMPLE PROBLEM

Compute the Eigenvalues and Eigenvectors of the following real symmetric matrix.

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 6 & 0 & 4 & 3 \\ 3 & 0 & 7 & 6 & 5 \\ 2 & 4 & 6 & 8 & 7 \\ 1 & 3 & 5 & 7 & 9 \end{bmatrix}$$

1. Goldstine, H. H., Murray, F. J., and VonNeuman, J. - The Jacobi Method for Real Symmetric Matrices. A. C. M. Journal, Volume 6, Number 1, 1959 pp 59-96.

2. Corbato, F. J. - On the Coding of Jacobi's Method for Computing Eigenvalues and Eigenvectors of Real Symmetric Matrices. A. C. M. Journal, Volume 10, Number 2. 1963. pp 123-125.

SAMPLE SOLUTION

```
*RUN EIGSR;EIG1
DØ YØU DESIRE USER INSTRUCTIØNS, TYPE YES ØR NØ
= YES
THIS PRØGRAM FINDS THE EIGENVALUES AND EIGENVECTØRS ØF A REAL
SYMMETRIC MATRIX BY THE JACØBI-CØRBATØ METHØD

THE MATRIX IS ØF THE FØRM
A11  A12  A1N      WHERE THE A(I,J) ARE REAL(FLØATING PØINT) AND
A21  A22  A2N      N(FIXED PØINT) CANNØT EXCEED 25:
AN1  AN2  ANN

SINCE THE MATRIX IS SYMMETRIC, ØNLY THE ELEMENTS
ØN AND ABØVE THE DIAGØNAL ARE INPUT.

THE PRØGRAM TYPES   A(1,1)=     THE USER TYPES THE FIRST RØW ELEMENTS.
    PRØGRAM TYPES   A(2,2)=     THE USER TYPES THE SECØND RØW,STARTING
    WITH THE DIAGØNAL ELEMENT.-ETC.-  TØ A(N,N)
INPUT IS TYPED IN THE FREE FIELD FØRMAT,A CARRIAGE RETURN ENDS THE FIELD

AFTER A(N,N) IS INPUT-THE PRØGRAM PRØVIDES THE ØPPØRTUNITY
AND INSTRUCTIØNS FØR CØRRECTING TYPING ERRØRS

NØW YØU TRY IT

ØRDER
= 5
A( 1, 1)
= 5,4,3,2,1
A( 2, 2)
= 6,0,4,3
A( 3, 3)
= 7,6,5
A( 4, 4)
= 8,8
A( 5, 5)
= 9
ARE ANY ØF THE ABØVE A(I,J) ELEMENTS TYPED INCØRRECTLY.
    IF USER WISHES TØ CØRRECT AN ELEMENT, TYPE YES, ØTHERWISE, TYPE NØ

ANY CØRRECTIØNS.
= YES
CØRRECT ELEMENT BY TYPING I SUBSCRIPT(RØW),SPACE ØR CØMMA,J SUBSCRIPT
(CØLUMN),SPACE ØR CØMMA,VALUE,CARRIAGE RETURN

= 4,7,7
ILLEGAL SUBSCRIPT, I GREATER THAN J,ØR I ØR J GREATER THAN N-TRY AGAIN

= 4,5,7
ANY CØRRECTIØNS.
= NØ
```

```
ØRDER=  5

      EIGENVALUE  1      EIGENVECTØR
      -1.0965953E+00       4.6935802E-01
                          -5.4221224E-01
                          -5.4445245E-01
                           4.2586562E-01
                           8.8988522E-02

      EIGENVALUE  2      EIGENVECTØR
       7.5137230E+00       5.5096188E-01
                           7.0944027E-01
                          -3.4017920E-01
                          -8.3410963E-02
                          -2.6543568E-01

      EIGENVALUE  3      EIGENVECTØR
       4.8489490E+00       5.4717278E-01
                          -3.1256994E-01
                           6.1811196E-01
                          -1.1560664E-01
                          -4.5549376E-01

      EIGENVALUE  4      EIGENVECTØR
       1.3270454E+00      -3.4101303E-01
                           1.1643460E-01
                           1.9590693E-02
                           6.8204296E-01
                          -6.3607129E-01

      EIGENVALUE  5      EIGENVECTØR
       2.2406871E+01       2.4587793E-01
                           3.0239601E-01
                           4.5321448E-01
                           5.7717706E-01
                           5.5638448E-01
THE SUM ØF THE SQUARES ØF THE ØFF DIAGØNAL ELEMENTS ØF

    XT A X =    2.6438681E-11                    WHERE XT = X-TRANSPØSE


ØRDER
= 0

PRØGRAM STØP AT 1460
*
```

ERRF

This FORTRAN function evaluates the error function or its complement.

INSTRUCTIONS

The calling sequence is:

Y = ERRF(X)

where,

- | X | is the independent variable
- Y is the error function, if X is greater than 0.
- Y is the complement of the error function, if X is less than 0.

RESTRICTIONS

If X = 0, the value returned is Y = 0.

For value of X greater than 13, exponential overflow may occur. For X not less than 6, a value of 1 is returned for the error function but the complementary error function is still significant.

SAMPLE PROBLEM

Find the value of the error function and its complement for the first 10 positive integers.

SAMPLE SOLUTION

```
10      PRINT 1
20    1 FØRMAT("          X              ERRF(X)        CØM ERRF(X)")
30      DØ 2 I=1,10
40      X=I
50      Y=ERRF(X)
60      Z=ERRF(-X)
70    2 PRINT 3,X,Y,Z
80    3 FØRMAT(1P3E16.7)
90      STØP;END

READY

*RUN *;ERRF
          X              ERRF(X)        CØM ERRF(X)
  1.0000000E+00   8.4270073E-01   1.5729926E-01
  2.0000000E+00   9.9532226E-01   4.6777331E-03
  3.0000000E+00   9.9997791E-01   2.2090515E-05
  4.0000000E+00   9.9999998E-01   1.5417261E-08
  5.0000000E+00   9.9999999E-01   1.5374581E-12
  6.0000000E+00   9.9999999E-01   2.1519699E-17
  7.0000000E+00   1.0000000E+00   4.1838189E-23
  8.0000000E+00   1.0000000E+00   1.1224288E-29
  9.0000000E+00   1.0000000E+00   4.1370322E-37
  1.0000000E+01   1.0000000E+00   0.

PRØGRAM STØP AT 90
*
```

ERRINV

This FORTRAN function finds the inverse of the error function.

INSTRUCTIONS

The calling sequence is:

Y = ERRINV(L, C)

where,

- Y is the inverse of the error function.
  When L = 0, C is the complement of the error function.
  When L = 1, C is the error function.

RESTRICTIONS

If the error function is input, a value of 6.0 will be returned for C greater than or equal to .999999997.

If the complement of the error is input, exponent overflow may occur when C is close to zero.

The library subprogram ERRF must be used with ERRINV as shown in the Sample Problem.

METHOD

Newton's iteration method is used.

SAMPLE PROBLEM

Find the inverse of the error function for values of .1, .2, .3, .4, .5, .6, .7, .8, and .9.

SAMPLE SOLUTION

```
10      PRINT 1
20    1 FØRMAT("          ERRF(X)                X")
30      DØ 2 I=1,9
40      C=I
50      C=C/10.0
60      Y=ERRINV(1,C)
70    2 PRINT 3,C,Y
80    3 FØRMAT(1P2E16.7)
90      STØP;END

READY

*RUN *;ERRINV;ERRF
       ERRF(X)              X
  1.0000000E-01    8.8856093E-02
  2.0000000E-01    1.7914334E-01
  3.0000000E-01    2.7246277E-01
  4.0000000E-01    3.7080727E-01
  5.0000000E-01    4.7693624E-01
  5.9999999E-01    5.9511594E-01
  7.0000000E-01    7.3286902E-01
  8.0000000E-01    9.0619390E-01
  9.0000000E-01    1.1630872E+00

PRØGRAM STØP AT 90
*
```

EUALG

This FORTRAN subprogram finds the greatest common divisor of two polynomials using a Euclidean algorithm.

INSTRUCTIONS

The calling sequence is

CALL EUALG (A, NA, B, NB, C, NC, TEST)

where

A	is the array of coefficients of the polynomial of greater order.
NA	is the degree of the polynomial in A.
B	is the array of coefficients of the polynomial of lesser order.
NB	is the degree of the polynomial in B.
C	is the coefficient array of the greatest common divisor polynomial.
NC	is the degree of the GCD polynomial.
TEST	is the criterion for determining zero coefficients of the polynomials. Proper functioning of the routine may depend on the judicious selection of this value.

The coefficients are stored in the A, B, and C arrays in order of increasing powers of X, i. e., A(1) = constant term.

RESTRICTIONS

$NB \leq NA \leq 25$

This routine calls the LIBRARY routine DVALG, which must be executed concurrently. See the sample solution.

SAMPLE PROBLEM

Find the GCD polynomial of

$$X^3 - 1$$

and

$$X^4 + X^3 + 2X^2 + X + 1$$

SAMPLE SOLUTION

```
*LIST

10 DIMENSION A(5),B(4),C(4)
20 DATA A,B/1.,1.,2.,1.,1.,-1.,0.,0.,1./
30 TEST=1E-4
40 CALL EUALG(A,4,B,3,C,NC,TEST)
50 PRINT 10,NC
60 10 FORMAT(" THE GCD POLYNOMIAL (DEGREE ",I2,")")
70 N=NC+1
80 DO 20 I=1,N
90 IM=I-1
100 20 PRINT 30,IM,C(I)
110 30 FORMAT(" THE COEF. OF X↑",I2,1PE20.7)
120 STOP
130 END

READY

*RUN *;EUALG;DVALG
THE GCD POLYNOMIAL (DEGREE  2)
THE COEF. OF X↑ 0        1.0000000E+00
THE COEF. OF X↑ 1        1.0000000E+00
THE COEF. OF X↑ 2        1.0000000E+00

PROGRAM STOP AT 120
*
```

FDRVUL

This FORTRAN function computes by numerical differentiation the first derivative of a tabulated function, where the independent variable may be unequally spaced.

INSTRUCTIONS

The calling sequence for the entry FDRVUL is:

ANS = FDRVUL(T,X,Y,NPTS,NORDER)

where,

- ANS is the value of the first derivative at T.
- T is the value of the independent variable at which the derivative is to be computed.
- X is a vector of tabulated values of the independent variable stored in ascending order.
- Y is a vector of tabulated values of the function corresponding to X.
- NPTS is the total number of paired (X,Y) points.
- NORDER is the desired order of the polynomial that will be used to approximate the first derivative.

RESTRICTION

Tabular values of the independent variable, X, must be stored in ascending order.

NORDER$<$NPTS

$X(1) \leq T \leq X(NPTS)$

METHOD

Lagrange's method for unequally spaced points is used. An NORDER polynomial is fit through the INT [(NORDER+1)/2] points preceding T and the (NORDER+1)-INT [(NORDER+1)/2] points following T. (INT denotes integer value.)

SAMPLE PROBLEM

Find the first derivative at T = 1.5 using a second degree polynomial and the following set of tabular data:

X	Y
0	0
1	1
2.5	6.25
4	16
5	25
6.5	42.25
7	49

SAMPLE SOLUTION

```
10        DIMENSIØN X(7),Y(7)
20        X(1)=0.0;Y(1)=0.0
30        X(2)=1.0;X(3)=2.5;X(4)=4.0;X(5)=5.0;X(6)=6.5;X(7)=7.0
40        Y(2)=1.0;Y(3)=6.25;Y(4)=16.0;Y(5)=25.0;Y(6)=42.25
50        Y(7)=49.0
60        T=1.5
70        NPTS=7
80        NØRDER=2
90        ANS=FDRVUL(T,X,Y,NPTS,NØRDER)
100        PRINT 30,T,ANS
110   30  FØRMAT(/"ØTHE VALUE ØF THE FIRST DERIVATIVE AT",F5.2," IS",
120&       1PE16.6)
130       STØP;END

READY

*RUN *;FDRVUL

THE VALUE ØF THE FIRST DERIVATIVE AT 1.50 IS     2.999999E+00

PRØGRAM STØP AT 130
*
```

FINT

This FORTRAN function evaluates by Fourier integral evaluation the function:

$$F_1(\omega) = \int_a^b f(x)\sin(\omega x)dx$$

or the function

$$F_2(\omega) = \int_a^b f(x)\cos(\omega x)dx$$

INSTRUCTIONS

The calling sequence for the entry FINT is:

Y = FINT(IND,A,B,N,W,FUNC)

where,

- Y is the value of the function.
- IND determines the function: IND = 1 $F_1(\omega)$ is obtained.
  IND = 2 $F_2(\omega)$ is obtained.
- A is the lower limit of integration.
- B is the upper limit of integration.
- N is the number of sample points to be used in the integration.
- W is a multiplier for the variable of integration in the argument of the sine or cosine term. W must be chosen so that ωx, where x is the variable of integration, is in radians.
- FUNC is the name of a function to supply values of f(x) given a value of x. An external statement must be used to define FUNC. For example, see the Sample Problem.

RESTRICTION

The term $W \cdot \frac{B-A}{N-1}$ must not be less than .007.

METHOD

Filon's formula is used to perform the integration.

If N is odd, the interval of integration is divided into equal sub-intervals of length H = (B-A)/(N-1) and the function f(x) is approximated by a parabola in each double interval.

If N is even, Filon's formula is used on the first N-1 points and in the remaining interval, f(x) is approximated by a linear interpolation*.

SAMPLE PROBLEM

Evaluate the function below, using W = 1 and 100 sample points, i.e., N = 100.

$$F_1(\omega) = \int_0^{15} e^{-x} \sin(x)dx$$

SAMPLE SOLUTION

```
10      EXTERNAL FX
20      Z=FINT(1,0.0,15.0,100,1.0,FX)
30      PRINT 10,Z
40   10 FORMAT(/24H VALUE OF THE FUNCTION =,1PE20.7)
50      STOP
60      END
70      FUNCTION FX(X)
80      FX=EXP(-X)
90      RETURN
100     END

READY

*RUN *,FINT

VALUE OF THE FUNCTION =        4.9999547E-01

PROGRAM STOP AT 50
*
```

* Hamming, R. W., Numerical Methods for Scientists and Engineers, McGraw-Hill, New York, 1962, Pages 319-321.

FRESNL

This FORTRAN subroutine evaluates the Fresnel integrals.

$$S(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \frac{\sin(t)}{\sqrt{t}}\, dt$$

and

$$C(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \frac{\cos(t)}{\sqrt{t}}\, dt$$

INSTRUCTIONS

The calling sequence is -

CALL FRESNL(SX, CX, X)

1. SX is the value of S(x)
2. CX is the value of C(x)
3. X is the real argument of S(x) and C(x).

METHOD

Boersma, "Computation of Fresnel Integrals", MTAC, V, 14, 1960, p. 380.

RESTRICTIONS

X, GE, 0. In the case of a negative argument the absolute value of X will be used.

SAMPLE PROBLEM

Find S(x) and C(x) for x = 1.5707963 and x = 6.2831853.

SAMPLE SOLUTION

```
*LIST

10 X=1.5707963
20 CALL FRESNL(SX,CX,X)
30 PRINT:"X,SX,CX",X,SX,CX
40 X=6.2831853
50 CALL FRESNL(SX,CX,X)
60 PRINT:"X,SX,CX",X,SX,CX
70 STØP
80 END

READY

*RUN *;FRESNL
X,SX,CX    1.5707963E+00    4.3825911E-01    7.7989335E-01
X,SX,CX    6.2831852E+00    3.4341568E-01    4.8825340E-01

PRØGRAM STØP AT 70
*
```

GAHER

This FORTRAN function performs Gauss-Hermite quadrature; i. e., it evaluates the integral

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx$$

INSTRUCTIONS

The calling sequence is-

ANS = GAHER(NARG, FUNC)

where,

ANS is the value of the integral.

NARG is the number of abscissae and weight coefficients to be used.

FUNC is the name of a function subprogram, supplied by the user, which evaluates only that part of the integrand represented by f(x). It is of the form-

FUNCTION FUNC(X) where X is the independent variable.

METHOD

See Kopal, Z., Numerical Analysis, (1961), p. 569.

RESTRICTIONS

1. 2. LE. NARG. LE. 20.

SAMPLE PROBLEM

Problem - To evaluate

$\int_{-\infty}^{\infty} e^{-x^2} \frac{x^2}{x^2+4} dx$ using the 12-point Gauss - Hermite quadrature, NARG = 12.

SAMPLE SOLUTION

```
*LIST

 10 EXTERNAL FUNC
 20 PRINT 10
 30 10 FØRMAT(" NARG     ANS")
 40 DO 30 NARG=2,20
 50 ANS=GAHER(NARG,FUNC)
 60 20 FØRMAT(1X,I2,3X,F12.8)
 70 30 PRINT 20,NARG,ANS
 80 STØP
 90 END
100 FUNCTION FUNC(X)
110 X2=X*X
120 FUNC=X2/(X2+4.)
130 RETURN
140 END

READY

*RUN *;GAHER
NARG     ANS
 2      0.19693931
 3      0.16113217
 4      0.16953905
 5      0.16721262
 6      0.16793688
 7      0.16769020
 8      0.16778045
 9      0.16763283
10      0.16775970
11      0.16775364
12      0.16775631
13      0.16775503
14      0.16775566
15      0.16775539
16      0.16775551
17      0.16775545
18      0.16775549
19      0.16775546
20      0.16775542

PRØGRAM STOP AT 80
*
```

GALA

This FORTRAN function performs Gauss-Laguerre quadrature, i. e., it evaluates the integral

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

INSTRUCTIONS

The calling sequence is-

ANS = GALA (NARG, FUNC)

1. ANS is the value of the integral.
2. NARG is the number of abscissae and weight coefficients to be used.
3. FUNC is the name of a function subprogram, supplied by the user, which evaluates only that part of the integrand represented by f(x). It is of the form

   FUNCTION FUNC(X)

   where X is the value of independent variable.

METHOD

See Kopal, Z., Numerical Analysis, (1961), p. 564.

RESTRICTIONS

1. 2. LE. NARG. LE. 15.

SAMPLE PROBLEM

Problem - To evaluate

$\int_0^\infty e^{-x} (x+4)^{-1} dx$ using the 7-point Gauss-Laguerre quadrature, NARG = 7.

SAMPLE SOLUTION

```
*LIST

10 EXTERNAL FUNC
20 PRINT 10
30 10 FORMAT(" NARG     ANS");
40 DO 30 NARG=2,15
50 ANS=GALA(NARG,FUNC)
60 20 FORMAT(1X,I2,3X,F12.8)
70 30 PRINT 20,NARG,ANS
80 STOP
90 END
100 FUNCTION FUNC(X)
110 FUNC=1./(X+4.)
120 RETURN
130 END

READY

*RUN *;GALA
NARG     ANS
 2      0.20588235
 3      0.20629370
 4      0.20633802
 5      0.20634429
 6      0.20634537
 7      0.20634558
 8      0.20634563
 9      0.20634563
10      0.20634564
11      0.20634564
12      0.20634564
13      0.20634564
14      0.20638337
15      0.20634564

PROGRAM STOP AT 80
*
```

GAMF

This FORTRAN function evaluates the gamma function.

INSTRUCTIONS

The calling sequence is:

Y = GAMF(X)

where,

- Y is the value of the gamma function.
- X is the value of the independent variable.

RESTRICTION

X may not be a negative integer or zero.

METHOD

Polynomial approximations.

SAMPLE PROBLEM

Find the gamma function for X = -1.5, -0.5, 0.5, 1.5.

SAMPLE SOLUTION

```
10        PRINT 1
15     1  FØRMAT("             X            GAMF(X)")
20        DØ 2 I=1,4
25        X=I
30        X=X-2.5
35        Y=GAMF(X)
40     2  PRINT 3,X,Y
45     3  FØRMAT(1P2E16.7)
50        STØP;END

READY

*RUN *;GAMF
            X               GAMF(X)
  -1.5000000E+00    2.3632720E+00
  -5.0000000E-01   -3.5449080E+00
   5.0000000E-01    1.7724540E+00
   1.5000000E+00    8.8622699E-01

PRØGRAM STØP AT 50
*
```

GAUSSN

This FORTRAN function evaluates a definite double or triple integral by Gaussian Quadrature.

INSTRUCTIONS

To evaluate the integral

(a) $\int_{A_1}^{B_1} \int_{A_2(w)}^{B_2(w)} \int_{A_3(v,w)}^{B_3(v,w)} f(u, v, w)\, du\, dv\, dw$

or

(b) $\int_{A_1}^{B_1} \int_{A_2(w)}^{B_2(w)} f(v, w)\, dv\, dw$

The calling sequence is

ANS = GAUSSN (M, N, FUNC)

where

1. M = 2 if the double integral (b) is to be solved
   M = 3 if the triple integral (a) is to be solved

2. N is the number of abscissae and weight coefficients to be used.

FUNC is the name of a function subprogram, supplied by the user, which evaluates the upper and lower limits of the integrals and the integrand. It is of the form:

FUNCTION FUNC (IGO, A, B, X)

where:

A, B, and X are vectors dimensioned A(M), B(M), and X(M)

X is the vector of the variables of integration, i.e., X(1) = w, X(2) = v, X(3) = u

If the IGO = M, the routine sets FUNC = f(x). If IGO < M, the routine sets A(IGO) = $A_{IGO}(X)$ and B(IGO) = $B_{IGO}(X)$.

ANS is the value of the integral.

METHOD

See Kopal, Z., Numerical Analysis, (1961), p. 386.

RESTRICTIONS

1. $2 \leq N \leq 16$
2. If $M \neq 2, 3$ then ANS = -10^{37}

SAMPLE PROBLEM

Integrate the triple integral

$$\int_0^1 \int_w^{2w} \int_{vw}^{2vw} uv \; du \; dv \; dw$$

Using a 7-point Gaussian method, where X(1) is used for w, X(2) is used for v, and X(3) is used for u. N = 7, M = 3.

SAMPLE SOLUTION

```
*
*LIST

10 EXTERNAL FUNC
20 NAMELIST /SOLN/N,ANS
30 N=7
40 M=3
50 ANS=GAUSSN(M,N,FUNC)
60 WRITE(" ",SOLN)
70 STOP
80 END
90 FUNCTION FUNC(IGO,A,B,X)
100 DIMENSION A(3),B(3),X(3)
110 GO TO (10,20,30,40),IGO
120 10 B(1)=1.
130 A(1)=0.
140 RETURN
150 20 B(2)=2.*X(1)
160 A(2)=X(1)
170 RETURN
180 30 B(3)=2.*X(2)*X(1)
190 A(3)=X(2)*X(1)
200 RETURN
210 40 FUNC=X(2)*X(3)
220 RETURN
230 END

READY

*RUN *;GAUSSN
NAMELIST        SOLN
N                    7
ANS          0.80357096E 00

PROGRAM STOP AT 70
*
```

GAUSSQ

This FORTRAN function computes the definite integral

$$\int_a^b f(t)dt$$

by Gaussian quadrature.

INSTRUCTIONS

The calling sequence for the entry GAUSSQ is:

Y = GAUSSQ(N,FUNC,A,B)

where:

- Y is the value of the integral.
- N is the number of values of f(t) to be used.
- FUNC is the name of a function that returns a value of f(t) given a value of t.
- A is the lower limit of the integral.
- B is the upper limit of the integral.

RESTRICTION

The number of values of f(t) must be between 2 and 8; i.e., $2 \leq N \leq 8$.

METHOD

For the method used in this subprogram, see Footnote[1].

SAMPLE PROBLEM

Evaluate by Gaussian quadrature the definite integral

$$\int_0^{2\pi} \sin(x)dx$$

using 6 values of the integrand. Use the value of 6.28 radians for 2π.

---

1 Scarborough, J. B., Numerical Mathematical Analysis, Johns Hopkins Press, Baltimore, Maryland, Third Edition, 1955, Article 54.

Note that in comparing the results of the two sample problems, CLCINT and GAUSSQ, the latter subprogram will obtain a more accurate result. The difference between these two results exists because of the mathematical method employed.

SAMPLE SOLUTION

```
10      EXTERNAL FUNC
20      Y=GAUSSQ(6,FUNC,0.,6.28)
30      PRINT 15, Y
40   15 FORMAT(/6X,23HVALUE OF THE INTEGRAL =,1PE20.7)
50      STOP
60      END
70      FUNCTION FUNC(X)
80      FUNC=SIN(X)
90      RETURN
100      END

READY

*RUN *;GAUSSQ

      VALUE OF THE INTEGRAL =        5.0337171E-06

PROGRAM STOP AT 50
*
```

GCDN

This FORTRAN subroutine finds the greatest common divisor of n integers a_i and multipliers z_i such that gcd = $z_1 a_1 + \ldots + z_n a_n$.

METHOD

Bradley's version of the Euclidean algorithm is used. [1] The number of arithmetic operations is linear in n.

INSTRUCTIONS

The calling sequence is

CALL GCDN (N, A, Z, IGCD)

where

N is the number of integers.

A is a single dimensioned integer array containing the input integers. The input is destroyed.

Z is a single dimensioned integer array used to output the N multipliers.

IGCD is the greatest common divisor of the A(I) integers.

SAMPLE PROBLEM

Find the GCD of -420, 0, 168, 252, 1260 and the multipliers z_i.

[1] Bradley, G. H., "Algorithm and Bound for the Greatest Common Divisor of n Integers", Communications of the ACM 13, p. 433 (1970).

SAMPLE SOLUTION

```
*LIST

10 INTEGER A(5),Z(5)
20 DATA A/-420,0,168,252,1260/
30 PRINT:"A=",A
40 CALL GCDN(5,A,Z,IGCD)
50 PRINT:"Z=",Z
60 PRINT:"GCD=",IGCD
70 STØP
80 END

READY

*RUN *;GCDN
A=          -420            0          168          252         1260
Z=             1            0           -2            0            0
GCD=            84

PRØGRAM STØP AT 70
*
```

GJSIMEQ

This FORTRAN subroutine solves a real system of N simultaneous equations in N unknowns of the form AX=B.

INSTRUCTIONS

The calling sequence is:

CALL SIMEQ(A, B, N, KERR, IDIM)

where,

- A is the name of a two dimensional array which contains the coefficient matrix in its first N rows and columns. It is destroyed during execution.
- B is the name of a one dimensional array containing the constant vector and which contains the solution vector on completion.
- N is an integer variable or constant which gives the order of the system.
- KERR is the name of a real variable which will be returned as a one if the system is singular, or a zero if the solution was found.
- IDIM is the dimension of A and B.

METHOD

The GAUSS-JORDAN method with pivotal condensation is used. A series of elementary row and column operations are applied to the matrix of coefficients, A, and to the constant vector, B. This process reduces the matrix of coefficients to the identity matrix. Then the resulting B array will contain the required solution.

RESTRICTIONS

This subroutine is best used for systems which have a dense matrix of coefficients, A. If a large sparse strongly diagonal matrix of coefficients is to be solved, then the GAUSS-SEIDEL iteration subroutine should be used.

SAMPLE PROBLEM

Solve the following systems of equations:

$$X_1 - 3X_3 + 7X_4 = 13$$

$$-4.5X_2 + 2X_3 + 13X_4 = 4$$

$$2X_1 + 2X_2 - X_3 = 7$$

$$-X_1 + X_2 + 9X_4 = 8$$

$$X_1 + 2X_2 + 3X_3 = 1$$

$$X_1 + 2X_2 + 3X_3 = 0$$

$$X_1 + X_3 = 3$$

SAMPLE SOLUTION

```
010       DIMENSIØN A(31,31),B(31)
020       KERR=0
030     1 PRINT:"ENTER THE ØRDER ØF THE SYSTEM"
040       READ:N
050       PRINT:"ENTER MATRIX CØLUMN BY CØLUMN"
060       READ:((A(I,J),I=1,N),J=1,N)
070       PRINT:"ENTER VECTØR ELEMENTS"
080       READ:(B(I),I=1,N)
090       CALL SIMEQ(A,B,N,KERR,31)
100       IF(KERR)55,52,55
110    52 PRINT:"THE SØLUTIØN IS"
120    60 PRINT:(B(I),I=1,N)
130       STØP
140    55 PRINT:"SINGULAR"
160       STØP:END

READY

*RUN *:GJSIMEQ
ENTER THE ØRDER ØF THE SYSTEM
= 4
ENTER MATRIX CØLUMN BY CØLUMN
= 1,0,-3,7
= 0,-4.5,2,13
= 2,2,-1,0
= -1,1,0,9
ENTER VECTØR ELEMENTS
= 13,4,7,8
THE SØLUTIØN IS
  -3.2672414E+00   2.5886699E+00   7.9790640E+00   -3.0911334E-01
PRØGRAM STØP AT 130
*RUN *:GJSIMEQ
ENTER THE ØRDER ØF THE SYSTEM
= 3
ENTER MATRIX CØLUMN BY CØLUMN
= 1,2,3
= 1,2,3
= 1,0,1
ENTER VECTØR ELEMENTS
= -1,0,3
SINGULAR

PRØGRAM STØP AT 160
*
```

GSEIDEL

This FORTRAN program solves a system of N simultaneous real equations in N unknowns, using the Gauss-Seidel iteration method.

INSTRUCTIONS

Supply data as requested.

GSEIDEL can also be used as a subroutine by deleting lines 1 through 669. The calling sequence for the entry would then be:

CALL GSEIDEL (A, B, N, C, IR)

where,

- A is the name of a two dimensional array containing the coefficients of the system of equations.
- B is the name of a one dimensional array containing the constant vector.
- N is the order of the system.
- X is the name of a one dimensional array which must contain initial estimate of the solution vector if IR is negative. X contains the solution vector on completion.
- IR is an integer variable giving the maximum number of iterations to be used to find the solution. If the users initial guess is to be used then IR should contain minus the number of iterations.

On return, IR equals the number of iterations needed to find the solution if one was found. If no solution was found, after the maximum number of iterations or the system was diverging, then IR is returned as zero. If one of the pivot elements is zero and the method cannot be used, then IR is returned as -1.

NOTE:

This method is used more effectively than the elimination method on a large sparce matrix of coefficients. This method is limited since it does not converge for all systems. However, for matrices which are strongly diagonal, convergence is assumed.

SAMPLE PROBLEM

Solve the following systems of equations:

$$X_1 - X_2 + X_3 + 5X_4 = -2$$

$$-X_1 + 2X_2 + 4X_3 + X_4 = 23$$

$$X_1 + 6X_2 + X_3 + 3X_4 = 11$$

$$4X_1 - 2X_2 + X_3 = 16$$

$$X_1 + X_2 + 3X_3 = 30$$

$$2X_1 + 5X_2 + X_3 = 211$$

$$5X_1 + 2X_2 + X_3 = 10$$

SAMPLE SOLUTION

```
*RUN

GSEIDEL

 ENTER ØRDER ØF SYSTEM (N):
= 4
ENTER CØEFFICIENT MATRIX RØW BY RØW:
= 1,-1,1,5
= -1,2,4,1
= 1,6,1,3
= 4,-2,1,0
 ENTER  CØNSTANT VECTØR
= -2,23,11,16
 ENTER  MAX. NØ. ØF ITERATIØNS
= -20
 ESTIMATED SØLUTIØN VECTØR ?
= -.5,3,8,2.7,3.2

 THE CØEFFICIENT MATRIX IS:

  1.0000000E+00  -1.0000000E+00   1.0000000E+00   5.0000000E+00 =
 -2.0000000E+00
 -1.0000000E+00   2.0000000E+00   4.0000000E+00   1.0000000E+00 =
  2.3000000E+01
  1.0000000E+00   6.0000000E+00   1.0000000E+00   3.0000000E+00 =
  1.1000000E+01
  4.0000000E+00  -2.0000000E+00   1.0000000E+00   0.            =
  1.6000000E+01
```

```
 THE SØLUTIØN VECTØR

   3.0438062E+00    1.2773663E+00    6.3795648E+00   -2.0292009E+00

 ANY ØTHER SYSTEMS TØ SØLVE ? (Y ØR N)
= Y

 ENTER ØRDER ØF SYSTEM (N):
= 3
ENTER CØEFFICIENT MATRIX RØW BY RØW:
= 1,1,3
= 2,5,1
= 5,2,1
 ENTER  CØNSTANT VECTØR
= 30,211,10
 ENTER  MAX. NØ. ØF ITERATIØNS
= 30

 THE CØEFFICIENT MATRIX IS:

   1.0000000E+00    1.0000000E+00    3.0000000E+00 =     3.0000000E+01
   2.0000000E+00    5.0000000E+00    1.0000000E+00 =     2.1100000E+02
   5.0000000E+00    2.0000000E+00    1.0000000E+00 =     1.0000000E+01

 THE SØLUTIØN VECTØR

  -1.7631583E+01    4.9368421E+01   -5.7894611E-01

 ANY ØTHER SYSTEMS TØ SØLVE ? (Y ØR N)
= N


PRØGRAM STØP AT 650
*
```

HDRVEB

This FORTRAN function computes the first, second, third, fourth, or fifth derivative of a tabulated function using difference quotients. The independent variable must be equally spaced.

INSTRUCTIONS

The calling sequence for the entry HDRVEB is:

ANS = HDRVEB(T,X,Y,NPTS,NDRV)

where,

- ANS is the value of the derivative requested at T.
- T is the value of the independent variable at which the derivative is to be computed.
- X is the name of the vector having the equally spaced tabulated values of the independent variable stored in ascending order.
- Y is the name of the vector of tabulated values of the function corresponding to X.
- NPTS is the total number of paired (X,Y) points.
- NDRV is defined as follows:

NDRV = 1 If the first derivative is desired
2 If the second derivative is desired
3 If the third derivative is desired
4 If the fourth derivative is desired
5 If the fifth derivative is desired

RESTRICTION

Tabular values of the independent variable, X, must be stored in ascending order and must be equally spaced.

T=X(J) for some J (J=1,2,. . . ,NPTS)

NPTS≥6

1≤NDRV≤5

METHOD

Six point numerical differentiation formulas are used. These formulas have an error term proportional to $h^6 \cdot f_6$ where h is the increment of the independent variable and f_6 is the sixth derivative of the function.

SAMPLE PROBLEM

Find the first five derivatives of the function $F(X) = X^5$ at T = 4.0 for X = 1, 2, 3,. . . . , 10.

SAMPLE SOLUTION

```
010       DIMENSIØN X(10),Y(10)
020       X(1)=0.0;Y(1)=0.0
030       DØ 10 I=2,10
040       X(I)=X(I-1)+1.0
050    10 Y(I)=X(I)**5
060       T=4.0
070       NPTS=10
090       DØ 20 I=1,5
100       NDRV=I
110       ANS=HDRVEB(T,X,Y,NPTS,NDRV)
120       PRINT 30,T,NDRV,ANS
130    30 FØRMAT("0FØR T=",F5.2," THE",I3," DERIVATIVE IS",1PE20.7)
150    20 CØNTINUE
160       STØP;END

READY

*RUN *;HDRVEB

FØR T= 4.00 THE  1 DERIVATIVE IS        1.2800000E+03

FØR T= 4.00 THE  2 DERIVATIVE IS        1.2800000E+03

FØR T= 4.00 THE  3 DERIVATIVE IS        9.6000000E+02

FØR T= 4.00 THE  4 DERIVATIVE IS        4.8000000E+02

FØR T= 4.00 THE  5 DERIVATIVE IS        1.2000000E+02

PRØGRAM STØP AT 160
*
```

JACELF

This FORTRAN subroutine computes the three Jacobian elliptic functions, sn, cn, and dn.

METHOD

Jacobian elliptic functions arise as inverse functions of elliptic integrals. Thus, if

$$x = \int_0^{\mu} \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}}$$

$$= \int_0^{\varphi} \frac{dt}{\sqrt{1 - k^2\sin^2 t}}$$

$= F(\varphi, k)$ = incomplete elliptic integral of first kind

Then

$$sn(x,k) = \sin\varphi = \mu$$
$$cn(x,k) = \cos\varphi = \sqrt{1 - \mu^2}$$
$$dn(x,k) = \sqrt{1 - k^2\sin^2\varphi} = \sqrt{1 - k^2\,\mu^2}$$

The value k is the modulus, c the complementary modulus, and s the square of the complementary modulus:

$$s = c^2 = 1 - k^2$$

INSTRUCTIONS

The calling sequence is -

CALL JACELF(SN, CN, DN, X, S)

where:

SN is the resultant value for sn(x, k).

CN is the resultant value for cn(x, k).

DN is the resultant value for dn(x, k).

X is the argument of the Jacobian elliptic functions.

S is the square of the complementary modulus.

REFERENCE

Bulirsch, R., Numerical Calculation of Elliptic Integrals and Elliptic Functions, Handbook Series of Special Functions, Numerische Mathematik, Vol. 7, 1965, pp. 78-90.

SAMPLE PROBLEM

Find SN, CN, and DN for x = .17475384, S = .75.

SAMPLE SOLUTION

```
*10 X=.17475384
*20 Y=.75
*30 CALL JACELF(SN,CN,DN,X,Y)
*40 PRINT:"SN,CN,DN",SN,CN,DN
*50 STØP
*60 END
*RUN *;JACELF
SN,CN,DN    1.7364817E-01    9.8480775E-01    9.9622364E-01

PRØGRAM STØP AT 50
*
```

LINEQ

This FORTRAN subroutine solves a system of simultaneous linear equations with real coefficients.

INSTRUCTIONS

The calling sequence is:

```
CALL LINEQ(A,B,NA,NB,IDIM)
```

where,

- A is the name of the array containing the matrix of coefficients. The coefficient matrix is altered during the course of the solution.
- B is the name of the array containing the right side vectors. The solutions are stored in B, thus destroying the original right hand vectors.
- NA is the number of equations.
- NB is the number of right side vectors.
- IDIM is the number of elements in the first dimension of the A and B arrays.

RESTRICTIONS

The first dimension of the A and B arrays must be IDIM, i.e., A(IDIM,I), B(IDIM,J).

The number of equations must not exceed 25.

METHOD

The solution is obtained by the Gaussian Elimination Method[1].

SAMPLE PROBLEM

Solve the following system of simultaneous linear equations for two right side set of constants.

```
   A - 2.*B + 3.*C + 4.*D = 4.5, 9.0
3.*A -    B + 2.*C + 5.*D = 9.5,19.0
2.*A + 4.*B - 5.*C +    D = 15.,30.0
4.*A + 2.*B -    C + 3.*D = 12.,24.0
```

[1]Scarborough, J. B., Numerical Mathematical Analysis, The Johns Hopkins Press, Sixth Edition, Baltimore, Maryland, 1963.

SAMPLE SOLUTION

```
10      DIMENSION A(25,4),B(25,2)
20      A(1,1)=1.0;A(3,4)=1.0
30      A(2,1)=3.0;A(1,3)=3.0;A(4,4)=3.0
40      A(3,1)=2.0;A(4,2)=2.0;A(2,3)=2.0
50      A(4,1)=4.0;A(3,2)=4.0;A(1,4)=4.0
60      A(1,2)=-2.0
70      A(2,2)=-1.0;A(4,3)=-1.0
80      A(3,3)=-5.0
90      A(2,4)=5.0
100       B(1,1)=4.5
110       B(2,1)=9.5
120       B(3,1)=15.0
130       B(4,1)=12.0
140       B(1,2)=9.0
150       B(2,2)=19.0
160       B(3,2)=30.0
170       B(4,2)=24.0
180       CALL LINEQ(A,B,4,2,25)
190       DO 5 I=1,2
200       PRINT 10, I
210   10  FORMAT(/19H SOLUTION TO VECTOR,I2)
220       PRINT 20, B(1,I),B(2,I)
230   20  FORMAT(3H A=,1PE20.7,5X,3H B=,1PE20.7)
240    5  PRINT 30, B(3,I),B(4,I)
250   30  FORMAT(3H C=,1PE20.7,5X,3H D=,1PE20.7)
260       STOP;END

READY

*RUN *;LINEQ

SOLUTION TO VECTOR 1
A=        -4.9999999E-01        B=         1.9999999E+00
C=        -1.0000000E+00        D=         3.0000000E+00

SOLUTION TO VECTOR 2
A=        -9.9999997E-01        B=         3.9999998E+00
C=        -2.0000001E+00        D=         6.0000000E+00

PROGRAM STOP AT 260
*
```

LINSR

This FORTRAN program solves a system of simultaneous linear equations of the form AX = B, where A is the coefficient matrix and B is the matrix of the right side vectors.

INSTRUCTIONS

Instructions for the format of the input data are generated by the program (see the sample solution). The order of the A matrix is requested initially then the A matrix is entered row-wise. The number of B vectors is then requested and the B matrix is entered column-wise. The program permits input errors to be corrected and generates the instructions required to make the corrections.

After corrections, if any, have been made, the program computes and prints the solution for each system accompanied by the error term. The error term is defined as:

$$E = \left[\frac{\sum_{i=1}^{N} (b_1 - \bar{b}_1)^2}{\sum_{i=1}^{N} (b_1)^2} \right] \quad 1/2$$

Where:

b_1 is the i^{th} component of the right side vector input by the user.
$\bar{b}_1$ is the i^{th} component of the right side vector computed by using the solution to the system.
After the solution to the system of equations provided by user has been generated, the user may define a new B matrix to be solved using the original A coefficient matrix.

METHOD

The method used to solve the systems of simultaneous linear equations is the Gaussian Elimination Method.

RESTRICTIONS

The order of the coefficient A matrix must be less than or equal to 25.

The maximum number of B vectors that can be solved at one time is 10.

NOTE

This program calls the subroutine LINEQ.

SAMPLE PROBLEM

Solve the following system of simultaneous linear equations:

$$2X_1 + 3X_2 = 12\ 1\ 3$$

$$X_1 - X_2 = 5\ 0\text{-}6$$

The matrix definitions for this system would be:

$$A = \begin{Bmatrix} 2 & 3 \\ 1 & -1 \end{Bmatrix} \quad X = \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix}, \text{ and } B = \begin{Bmatrix} 12\ 1 & 3 \\ 5\ 0 & -6 \end{Bmatrix}$$

SAMPLE SOLUTION

```
*RUN LINSR;LINEQ
DØ YØU DESIRE USER INSTRUCTIØNS, TYPE YES ØR NØ
= YES
THIS RØUTINE SØLVES A SYSTEM ØF SIMULTANEØUS LINEAR EQUATIØNS ØF THE
FØRM AX=B. THE METHØD USED IS GAUSSIAN ELIMINATIØN. A IS THE NAME ØF
THE CØEFFICIENT MATRIX ENTERED RØW-WISE. THE ØRDER ØF THE SYSTEM,
N, CANNØT EXCEED 25.
THE B ARRAY IS A MATRIX ØF THE RIGHT HAND SIDE VECTØRS ENTERED CØLUMN-
WISE. THE PRØGRAM SØLVES A SYSTEM ØF 10 ØR FEWER VECTØRS, AFTER WHICH
IT REQUESTS MØRE RIGHT HAND SIDE VECTØRS.
INPUT FØRMAT IS FREE FIELD: EACH NUMBER IS FØLLØWED BY A BLANK ØR CØMMA.
A(I,J) AND B(I,J) ARE REAL. ØRDER AND NUMBER ØF VECTØRS ARE INTEGER.

FØR EQUATIØNS   3*X1 + 7*X2 + .5*X3 = 10  4

               -2*X1 +       + 10*X3 =  5  2

                1*X1 +.3*X2 + 17*X3 =  2 .4

INPUT IS AS FØLLØWS:

ØRDER=3

A(1,1)=3.,7.,.5

A(2,1)=-2.,0.,10.

A(3,1)=1.,-3.E-1,17.

NUMBER ØF RIGHT VECTØRS=2

B(1,1)=10.,5.,2.

B(1,2)=4.,2.,.4

NØW YØU TRY IT

ØRDER
= 2
A( 1,1)
= 2.,3.
A( 2,1)
= 1.,1.
ANY CØRRECTIØNS, TYPE YES ØR NØ
= YES
```

```
TYPE RØW, CØLUMN, AND CØRRECTED ELEMENT
E.G.-TØ CØRRECT A(10,3)=15., TYPE 10,3,15.

= 2,2,-1.
ANY CØRRECTIØNS, TYPE YES ØR NØ
= NØ
NUMBER ØF RIGHT VECTØRS
= 3
B(1, 1)
= 12.,5.
B(1, 2)
= 1.,0.
B(1, 3)
= 3.,6.
ANY CØRRECTIØNS, TYPE YES ØR NØ
= YES
TYPE RØW, CØLUMN, AND CØRRECTED ELEMENT

= 3,3,-6.
ILLEGAL SUBSCRIPT, I ØR J GREATER THAN N TRY AGAIN

= 2,3,-6.
ANY CØRRECTIØNS, TYPE YES ØR NØ
= NØ
     SØLUTIØN
     0.54000000E+01
     0.40000000E+00
THE RELATIVE ERRØR IS       0.10252311E-07

     SØLUTIØN
     0.20000000E+00
     0.20000000E+00
THE RELATIVE ERRØR IS       0.18626451E-08

     SØLUTIØN
    -0.30000000E+01
     0.30000000E+01
THE RELATIVE ERRØR IS       0.

THE CØEFFICIENT MATRIX HAS BEEN SAVED. DØ YØU HAVE ANY MØRE
RIGHT HAND VECTØRS TØ SØLVE, TYPE YES ØR NØ
= NØ

PRØGRAM STØP AT 0
*
```

MTALG

This FORTRAN subroutine finds the product of two polynomials.

INSTRUCTIONS

The calling sequence for the entry MTALG is:

CALL MTALG(A,NA,B,NB,C)

where,

- A is the name of the multiplicand coefficient array.
- NA is the degree of the multiplicand polynomial.
- B is the name of the multiplier coefficient array.
- NB is the degree of the multiplier polynomial.
- C is the name of the product coefficient array.

All polynomial coefficients are stored constant term first.

RESTRICTION

Neither NA nor NB may exceed 25.

SAMPLE PROBLEM

Find the product of the polynomial $X^2 - 3$ multiplied by the polynomial $X^5 + 2X-5$.

SAMPLE SOLUTION

```
LIST

010      DIMENSION A(3),B(6),C(8)
020      A(1)=-3.0
030      B(1)=-5.0
040      B(2)=2.0
050      B(3)=0.0; B(4)=0.0; B(5)=0.0; A(2)=0.0
060      B(6)=1.0; A(3)=1.0
070      CALL MTALG(A,2,B,5,C)
080      PRINT 10
090   10 FORMAT(/21H PRODUCT COEFFICIENTS)
100      DO 15 I=1,8
110      M=I-1
120   15 PRINT 20, M,C(I)
130   20 FORMAT(/19H COEFFICIENT OF X**,I2,1X,E20.7)
140      STOP
150      END

READY

*RUN *;MTALG

PRODUCT COEFFICIENTS

COEFFICIENT OF X** 0          0.1500000E+02

COEFFICIENT OF X** 1         -0.6000000E+01

COEFFICIENT OF X** 2         -0.5000000E+01

COEFFICIENT OF X** 3          0.2000000E+01

COEFFICIENT OF X** 4          0.

COEFFICIENT OF X** 5         -0.3000000E+01

COEFFICIENT OF X** 6          0.

COEFFICIENT OF X** 7          0.1000000E+01

PROGRAM STOP AT 140
*
```

MTINV

This FORTRAN subprogram inverts a matrix and/or solves linear systems by standard elimination.

INSTRUCTIONS

The calling sequence is as follows:

```
CALL MTINV (A, NR, NC, IDIM, LABEL)
```

where:

A	=	single precision matrix array
NR	=	number of rows
NC	=	number of columns
IDIM	=	first dimension of A, i.e., A(IDIM, T)
LABEL	=	scratch array containing at least NR cells.

The routine will invert the NR x NR matrix A in place, and will treat any additional columns as right-hand sides of a system. The solutions will be returned in the corresponding columns.

RESTRICTION

The determinant of A must not be zero.

SAMPLE PROBLEM

Invert the matrix

$$A = \begin{bmatrix} 1 & -2 & 3 & 4 \\ 3 & -1 & 2 & 5 \\ 2 & 4 & -5 & 1 \\ 4 & 2 & -1 & 3 \end{bmatrix}$$

and also solve the systems

$$Ax = (1 \quad 0 \quad 0 \quad 0)^T$$

and

$$Ax = (1 \quad 1 \quad 1 \quad 1)^T$$

SAMPLE SOLUTION

```
10 DIMENSIØN A(10,10),LABEL(10)
20 PRINT:"SIZE ØF MATRIX";READ:IDIM
30 PRINT:"MATRIX BY RØWS"
40 READ:((A(I,J),J=1,IDIM),I=1,IDIM)
50 PRINT:"# ØF RHS"
60 READ:KRHS
70 IEND=IDIM+KRHS
80 IF(KRHS.LE.0)GØ TØ 1
90 PRINT:"ENTER RHS'S"
100 ISTR=IDIM+1
110 READ:((A(J,I),J=1,IDIM),I=ISTR,IEND)
120 1 CALL MTINV(A,IDIM,IEND,10,LABEL)
130 PRINT:"INVERTED MATRIX"
140 PRINT:((A(I,J),J=1,IDIM),I=1,IDIM)
150 IF(KRHS.LE.0)STØP
160 PRINT:"RHS'S"
170 PRINT:((A(J,I),J=1,IDIM),I=ISTR,IEND)
180 STØP;END

READY

*RUN *;MTINV
SIZE ØF MATRIX
= 4
MATRIX BY RØWS
= 1 -2  3  4
= 3 -1  2  5
= 2  4 -5  1
= 4  2 -1  3
# ØF RHS
= 2
ENTER RHS'S
= 1 0 0 0
= 1 1 1 1
INVERTED MATRIX
  -1.2000000E+00   1.1000000E+00  -2.5000000E-01  -1.5000000E-01
   2.2000000E+00  -2.6000000E+00   0.              1.4000000E+00
   1.4000000E+00  -1.7000000E+00  -2.5000000E-01   1.0500000E+00
   6.0000000E-01  -3.0000000E-01   2.5000000E-01  -5.0000002E-02
RHS'S
  -1.2000000E+00   2.2000000E+00   1.4000000E+00   6.0000000E-01
  -4.9999999E-01   9.9999996E-01   4.9999997E-01   4.9999999E-01

PRØGRAM STØP AT 180
*
```

MTMPY

This FORTRAN subroutine evaluates the product of two matrices.

INSTRUCTIONS

The calling sequence for this subprogram is:

CALL MTMPY (IND,A,B,C,L,M,N)

where,

IND	DESCRIPTION	INDICES	EXAMPLE
0	$A \times B = C$	L,M	A(L,M) x B(M,N) = C(L,N)
	$A^T \times B = C$	-L,M	A^T(L,M) x B(M,N) = C(L,N)
	$A \times B^T = C$	L,-M	A(L,M) x B^T(M,N) = C(L,N)
	$A^T \times B^T = C$	-L,-M	A^T(L,M) x B^T(M,N) = C(L,N)
1	$D[A] \times D[B] = C$	L,L	D[A(L,L)] x D[B(L,L)] = D[C(L,L)]
2	$D[A] \times B = C$	L,M	D[A(L,L)] x B(L,M) = C(L,M)
	$D[A] \times B^T = C$	L,-M	D[A(L,L)] x B^T(L,M) = C(L,M)
3	$A \times D[B] = C$	L,M	A(L,M) x D[B(M,M)] = C(L,M)
	$A^T \times D[B] = C$	-L,M	A^T(L,M) x D[B(M,M)] = C(L,M)

- A is the name of the multiplicand matrix.
- B is the name of the multiplier matrix.
- C is the name of the product matrix.
- L, M, and N are as shown above.

To transpose a matrix, L and/or M must be input as negative values.

A^T means the transpose of A.

D[A] means the diagonal of A.

RESTRICTIONS

The first dimension of the A, B, and C arrays must be 25, i.e., A(25,I), B(25,J), C(25,K).

Each matrix has a maximum of 25 rows.

For the following cases, the matrices are restricted to a maximum of 25 columns.

IND	INDICES	MATRIX
0	-L,M	A
0	-L,-M	A and B
2	L,-M	B
3	-L,M	A

SAMPLE PROBLEM

Multiply matrix A by the transpose of matrix B. The matrices are defined as follows.

$$A = \begin{bmatrix} 3. & 2. & 1. \\ 1. & 2. & 1. \\ 4. & 1. & 0. \end{bmatrix} \qquad B = \begin{bmatrix} 2. & 1. & 4. \\ 1. & 3. & 1. \end{bmatrix}$$

SAMPLE SOLUTION

```
*LIST

10        DIMENSION A(25,3),B(25,3),C(25,2)
20        A(1,1)=3.0;B(1,2)=3.0
30        A(2,1)=1.0;A(3,2)=1.0;A(1,3)=1.0;A(2,3)=1.0
40        B(1,1)=1.0;B(2,2)=1.0;B(1,3)=1.0
50        A(3,1)=4.0;B(2,3)=4.0
60        A(1,2)=2.0;A(2,2)=2.0;B(2,1)=2.0
70        A(3,3)=0.0
80        CALL MTMPY(0,A,B,C,3,-3,2)
90        PRINT 10
100    10 FORMAT("              MATRIX       *  TRANSPOSE MATRIX")
110       DO 20 I=1,3
120    20 PRINT 30, (C(I,J),J=1,2)
130    30 FORMAT(1P2E20.7)
140       STOP;END

READY

*RUN *;MTMPY
          MATRIX       *  TRANSPOSE MATRIX
      1.0000000E+01        1.2000000E+01
      8.0000000E+00        8.0000000E+00
      7.0000000E+00        9.0000000E+00

PROGRAM STOP AT 140
*
```

NCOATES

This FORTRAN program evaluates the integral of a function over a closed interval by the Newton-Coates closed interval method.

METHOD

The user has the choice of using any of the two through ten point Newton-Coates closed interval formulas including the well known:

Trapezoidal Rule	=	2 point
Simpson's Rule	=	3 point
3/8 Rule	=	4 point
Bode's Rule	=	5 point

If the interval is not broken down into an even number of steps for the particular point formula used, then an appropriate point formula of lesser degree is used to complete the last step.

INSTRUCTIONS

To use this program enter data as requested. For more instructions run the program.

NCOATES may also be used as a subroutine, by deleting lines 1 through 999. The calling sequence for entry NCOATES would then be:

CALL NCOATES (RESULT, VECTOR, NPOINT, NTOT, H)

where,

- RESULT is the name of a real variable which is to receive the value of the integral.
- VECTOR is a one dimensional array containing the values of the function at successive internal points.
- NPOINT is an integer variable or constant from 2 to 10 indicating the desired Newton Coates formula to be used.
- NTOT is an integer number of functional values in VECTOR.
- H is a real variable or constant giving the step size.

NOTE:

An H-point method should integrate a polynomial of degree up to n + 1, without error. However, high N-point formulas (NPOINT > 8) are rarely used unless knowledge of higher order derivatives is known, because the subtractions lead to a loss of accuracy. For the commonly used Simpson's Rule the error depends on the fourth derivative of the function and is given by $\frac{(b-a)^5}{90} * D^4 f(\varepsilon)$ for some ε in $[a, b]$.

One will also receive more accurate results if the point formula and total number of points are chosen such that the last step does not have to be completed by a lower order formula.

SAMPLE PROBLEM

Evaluate the integral from 0 to 1 of sin (X).

SAMPLE SOLUTION

```
*10 FUNCTION FUNC (X)
*20 FUNC = SIN (X)
*30 RETURN
*40 END
*RUN

NCOATES


WOULD YOU LIKE INSTRUCTIONS?  YES OR NO
= NO

N-POINT FORMULA, INPUT N
= 2
NUMBER OF STEPS
= 5
INTEGRATION LIMITS
= 0,1
INTEGRAL =     4.5816433E-01

N-POINT FORMULA, INPUT N
= 3
NUMBER OF STEPS
= 5
INTEGRATION LIMITS
= 0,1
INTEGRAL =     4.5917869E-01

N-POINT FORMULA, INPUT N
= 5
NUMBER OF STEPS
= 5
INTEGRATION LIMITS
= 0,1
INTEGRAL =     5.0381142E-01

N-POINT FORMULA, INPUT N
= 0

PROGRAM STOP AT 580
*
```

NUMINT

This BASIC program evaluates definite integrals using the Gaussian quadrature using ten values of the integrand.

INSTRUCTIONS

To use this program, enter the integrand as follows:

```
100 LET Y = A FUNCTION OF X
```

Then type "RUN"

When RUN has been typed the program will ask for the integration limits:

```
L, U, N =
```

Where L = Lower limit
U = Upper limit
N = Number of intervals

If more than one integral of the same function is to be evaluated, provide the limits each time the question is typed. To end the program, equate the upper and lower limits.

NOTE:

Lines 100 thru 129 can be used to express the function.

SAMPLE PROBLEM

Find the value of the following integral between the limits 0 to 7.65 and 7.65 to 1567.

$$\int \frac{X^2}{\log 3X} dX$$

To evaluate the integrand, enter the program line:

```
100 LET Y = X ↑ 2/(LOG(3*X)/2.30258509)
```

The division by 2.30258509 is necessary to convert from natural to common logarithms.

SAMPLE SOLUTION

```
*100 LET Y=X↑2/(LØG(3*X)/2.30258509)
*RUN


L, U, N = ?0, 7.65, 1

      THE INTEGRAL FRØM      0   TØ   7.65   =      125.5347

L, U, N = ?7.65, 1567, 1

      THE INTEGRAL FRØM   7.65   TØ   1567   =      3.64278 E 8

L, U, N = ?0, 0, 1

READY
*
```

ORTHP

This FORTRAN subprogram evaluates an orthogonal polynomial.

INSTRUCTIONS

The calling sequence for the entry ORTHP is:

```
Y = ORTHP(IND, Z, N)
```

where,

- Y is the value of the orthogonal polynomial.
- IND is the type of orthogonal polynomial generated-

  1. Legendre (of the first kind)
  2. Laguerre
  3. Hermite
  4. Chebychev

- Z is the value of the independent variable.
- N is the degree of the polynomial.

RESTRICTIONS

For orthogonality:

Legendre-	|Z| less than or equal to 1.0
Laguerre-	Z equal to or greater than 0
Hermite-	none.
Chebychev-	|Z| less than or equal to 1.0

METHOD

Legendre-

```
P(0)=1.0
P(1)=Z
P(N+1)=((2.0*N+1.0)*Z*P(N)-N*P(N-1))/(N+1.0)
```

Laguerre-

```
L(0)=1.0
L(1)=1.0-Z
L(N+1)=((1.0+2.0*N-Z)*L(N)-N*L(N-1))/(N+1.0)
```

Hermite-

H(0)=1. 0
H(1)=2. 0*Z
H(N+1)=2. 0*Z*H(N)-2. 0*N*H(N-1)

Chebychev-

T(0)=1. 0
T(1)=Z
T(N+1)=2. 0*Z*T(N)-T(N-1

SAMPLE PROBLEM

Find the values of the following:

Legendre polynomial for N=9 and Z=0. 5.
Laguerre polynomial for N=9 and Z=10. 0.
Hermite polynomial for N=3 and Z=1. 0.
Chebychev polynomial for N=5 and Z=0. 6.

SAMPLE SOLUTION

```
*LIST

110       IND=1
120       Z=0.5
130       N=9
140       Y=ØRTHP(IND,Z,N)
150       PRINT 10,N,Z,Y
160    10 FØRMAT(/" FØR N=",I2," AND Z=",F5.2/" THE VALUE ØF THE ",
170&      "LEGENDRE PØLYNØMIAL IS",1PE14.6)
180       IND=2
190       Z=10.0
200       N=9
210       Y=ØRTHP(IND,Z,N)
220       PRINT 20,N,Z,Y
230    20 FØRMAT(/7H FØR N=,I2,7H AND Z=,F5.2,/,40H THE VALUE ØF HTE
240& LAGUERRE PØLYNØMIAL IS,1PE14.6)
250       N=3
260       IND=3
265        Z=1.0
270       Y=ØRTHP(IND,Z,N)
280       PRINT 30,N,Z,Y
290    30 FØRMAT(/7H FØR N=,I2,7H AND Z=,F5.2,/,39H THE VALUE ØF THE
300& HERMITE PØLYNØMIAL IS, 1PE14.6)
310       IND=4
320       Z=0.6
330       N=5
340       Y=ØRTHP(IND,Z,N)
350       PRINT 40,N,Z,Y
360    40 FØRMAT(/7H FØR N=,I2,7H AND Z=,F5.2,/,41H THE VALUE ØF THE
370& CHEBYCHEV PØLYNØMIAL IS, 1PE14.6)
380       STØP
390       END

READY

*RUN *ØRTHP

FØR N= 9 AND Z= 0.50
THE VALUE ØF THE LEGENDRE PØLYNØMIAL IS -2.678986E-01

FØR N= 9 AND Z=10.00
THE VALUE ØF HTE LAGUERRE PØLYNØMIAL IS  1.479189E+01

FØR N= 3 AND Z= 1.00
THE VALUE ØF THE HERMITE PØLYNØMIAL IS -4.000000E+00

FØR N= 5 AND Z= 0.60
THE VALUE ØF THE CHEBYCHEV PØLYNØMIAL IS -7.583999E-02

PRØGRAM STØP AT 380
*
```

PLMLT

This FORTRAN subroutine constructs the coefficients of a polynomial from its real roots.

INSTRUCTIONS

The calling sequence for the entry PLMLT is:

```
CALL PLMLT(Z,NZ,P)
```

where,

- Z is the name of the array containing the roots.
- NZ is the number of roots.
- P is the name of the coefficient array. It is dimensioned at least NZ+1 locations. The coefficients are stored low order first and the coefficient P(NZ+1)=1.0.

SAMPLE PROBLEM

Construct the coefficients of the polynomial whose factored form is (X-2) (X+2) = 0.

SAMPLE SOLUTION

```
10      DIMENSION P(3),Z(2)
20      Z(1)=2.0
30      Z(2)=-2.0
40      NZ=2
50      CALL PLMLT(Z,NZ,P)
60      DO 10 I=1,3
70      M=I-1
75   10 PRINT 20,M,P(I)
80   20 FORMAT(/19H COEFFICIENT OF X**,I2,1PE20.7)
90      STOP;END

READY

*RUN *;PLMLT

COEFFICIENT OF X** 0       -4.0000000E+00

COEFFICIENT OF X** 1        0.

COEFFICIENT OF X** 2        1.0000000E+00

PROGRAM STOP AT 90
*
```

POLRTS

This FORTRAN program finds the roots of polynomials using Bairstow's method.

INSTRUCTIONS

POLRTS is written to allow a 30th degree polynomial. The polynomial itself must be written in descending powers of the independent variable:

$$C_1Z^n + C_2Z^{n-1} + \ldots + C_nZ + C_{n+1}$$

In addition, the polynomial must have no 0 roots; that is, the coefficient C_{n+1} must not be 0. If C_{n+1} is 0, the polynomial must be rewritten as a polynomial of degree m:

$$C_1Z^m + C_2Z^{m-1} + \ldots + C_mZ + C_{m+1}$$

where m = n-1. If C_{m+1} is 0, this must be repeated.

RESTRICTIONS

This routine may not give satisfactory results for certain ill-conditioned polynomials or polynomials having multiple roots.

SAMPLE PROBLEM

Determine the roots for the following polynomial:

$$1.5X^7 + 2.906X^6 + 10.6X^5 + 25.877X^4 + 2.3X^3 + 33X^2 + 1.234X + 543.2 = 0$$

SAMPLE SOLUTION

```
*RUN
ENTER 'NX' THE DEGREE ØF THE PØLYNØMIAL
= 7
ENTER 'NC' AND 'C' THE NUMBER ØF CØEFFICIENTS AND
THE LIST ØF NX+1 CØEFFICIENTS (DESCENDING)
= 8  1.5  2.906  10.6  25.877  2.3  33  1.234  543.2

   PØLYNØMIAL TØ BE SØLVED IS

          1.500000E+00X**7

          2.906000E+00X**6

          1.060000E+01X**5

          2.587700E+01X**4

          2.300000E+00X**3

          3.300000E+01X**2

          1.234000E+00X

          5.432000E+02


     RØØTS ARE AS FØLLØWS

            REAL PØRTIØN    IMAGINARY PØRTIØN

          1   2.137941E-01    2.873626E+00I

          2   2.137941E-01   -2.873626E+00I

          3   1.441992E+00    1.153711E+00I

          4   1.441992E+00   -1.153711E+00I

          5  -1.244203E+00    1.756273E+00I

          6  -1.244203E+00   -1.756273E+00I

          7  -2.760499E+00    0.           I
DØ YØU WANT TØ FIND THE RØØTS ØF ANØTHER PØLYNØMIAL?
ANSWER 1 FØR YES ØR 0 FØR NØ
= 0

PRØGRAM STØP AT 520
*
```

POLYC

This FORTRAN subprogram constructs the coefficients of a real polynomial from its real and/or complex conjugate roots.

INSTRUCTIONS

The calling sequence for the entry POLYC is:

```
CALL POLYC(R, C, N, A)
```

where,

- R is the array containing the real roots and/or the real parts of the complex conjugate pairs.
- C is the array containing the imaginary parts of the complex conjugate pairs, and zeros corresponding to the real roots.
- N is the number of roots.
- A is the polynomial coefficient array.

RESTRICTIONS

N must be less than or equal to 25.

If there are complex conjugate roots, only one of the conjugates need be input. However, an adjacent location must be saved for the other conjugate. For example, for the roots 1+I, 1-I, 3+0I, input is:

```
R(1) = 1
C(1) = 1
R(3) = 3
C(3) = 0
```

SAMPLE PROBLEM

Construct the coefficients of the real polynomial whose roots are:

```
Root 1 = -1-I
Root 2 = -1+I
Root 3 = -1
Root 4 =   -I
Root 5 =   +I
```

SAMPLE SOLUTION

```
*LIST

10 DIMENSION R(5),C(5),A(6)
20 DATA C/-1.,1.,0.,-1.,1./,R/-1.,-1.,-1.,0.,0./,N/5/
30 CALL POLYC(R,C,N,A)
40 DØ 10 I=1,6
50 10 PRINT 20,I-1,A(I)
60 20 FØRMAT("0CØEFFICIENT ØF X**",I2,1PE20.7)
70 STØP;END

READY

*RUN *;PØLYC

CØEFFICIENT ØF X** 0        2.0000000E+00

CØEFFICIENT ØF X** 1        4.0000000E+00

CØEFFICIENT ØF X** 2        5.0000000E+00

CØEFFICIENT ØF X** 3        5.0000000E+00

CØEFFICIENT ØF X** 4        2.0000000E+00

CØEFFICIENT ØF X** 5        0.0000000E+00

PRØGRAM STØP AT 70
*
```

POLYV

This FORTRAN subroutine evaluates a real polynomial with a complex argument expressed in either polar or Cartesian coordinates.

INSTRUCTIONS

The calling sequence for the entry POLYV is:

```
CALL POLYV(N,A,RHO,PHI,IND,R,C)
```

where,

- N is the degree of the polynomial.
- A is the polynomial coefficient array with the coefficients entered low order first.
- RHO is the length of the radius vector in polar coordinates, the real part of the Cartesian argument.
- PHI is the polar angle in radians for polar coordinates, the imaginary part of the Cartesian argument.
- IND determines the coordinate system.

  IND=0 polar coordinates
  IND=1 Cartesian coordinates

- R is the value of the real part of the polynomial evaluation.
- C is the value of the imaginary part of the polynomial evaluation.

SAMPLE PROBLEM

Evaluate the following polynomial for a complex argument whose polar coordinates are length of radius vector = 2.0 and polar angle = 0.0 radians:

$$X^6 + X^4 - X^2 - 1.0 = 0$$

SAMPLE SOLUTION

```
10      DIMENSION A(7)
20      A(1)=-1.0;A(3)=-1.0
30      A(5)=1.0;A(7)=1.0
40      A(2)=0.0;A(4)=0.0;A(6)=0.0
50      N=6
60      IND0=0
70      CALL POLYV(N,A,2.0,0.0,IND,R,C)
80      PRINT 10, R
90      PRINT 20, C
100   10 FORMAT(/23H VALUE OF THE REAL PART,1PE25.7)
110   20 FORMAT(/28H VALUE OF THE IMAGINARY PART, 1PE20.7)
120     STOP;END

READY

*RUN *;POLYV

VALUE OF THE REAL PART              7.4999996E+01

VALUE OF THE IMAGINARY PART         0.

PROGRAM STOP AT 120
*
```

QUADEQ

This BASIC program finds the solution to quadratic equations.

INSTRUCTIONS

To use this program, provide the coefficients of the quadratic equation per the standard mathematical form,

$$ax^2 + bx + c = 0$$

when requested by the program.

SAMPLE PROBLEM

Solve the equations:

1) $3x^2 + x + 5 = 0$

2) $x^2 + 4x - 6 = 0$

3) $1E25\ x^2 - 2E28x + 1E25 = 0$

SAMPLE SOLUTION

```
*RUN

QUADEQ
I SØLVE THE QUADRATIC EQ.  A*X*X+B*X+C=0
INPUT A,B,C ?3,1,5
CØMPLEX RØØTS:  -.1666667 (+ AND -)   1.280191  I

MØRE EQ'S TØ SØLVE (1=YES, 0=NØ) ?1
INPUT A,B,C ?1,4,-6
REAL RØØTS:  -5.162278  AND    1.162278

MØRE EQ'S TØ SØLVE (1=YES, 0=NØ) ?1
INPUT A,B,C ?1E25,-2E28,1E25
REAL RØØTS:   1999.999  AND    .0005

MØRE EQ'S TØ SØLVE (1=YES, 0=NØ) ?0

READY
*
```

RKPBX

This FORTRAN subprogram contains two routines, RKPB1 and RKPB2 to integrate systems of first-order ordinary differential equations by the fourth-order Runge-Kutta method.

INSTRUCTIONS

There are two entries to this subprogram. They are:

```
CALL RKPB1(DERIV,TEMP,X,DX,Y,F,N)

CALL RKPB2(DERIV,TEMP,X,DX,Y,F,N)
```

where,

- DERIV is the name of the routine which computes values for the derivatives given values for the independent variable and the dependent variables. An external statement must be entered to define DERIV, as shown in the Sample Problem.
- TEMP is the name of an array containing at least 4*(N+1) locations which must not be used for any other purpose while the integration is being performed.
- X is the value of the independent variable.
- DX is the value of the independent increment.
- Y is the name of the array containing the dependent variables.
- F is the name of the array containing the dependent derivatives.
- N is the number of dependent variables.

METHOD*

RKPB1 will compute the derivatives and store the functions and derivatives at each step of the integration. RKPB2 will integrate to the next step. The values of the functions and derivatives at the next step will be stored. However, the integration may be repeated with an adjusted increment. The integration step will be made permanent only by calling RKPB1.

The routine is designed only to perform the integration of differential equations. Provision for output, termination of the integration, and adjustment of the increment must be made by the user. Generally, the output and termination should be done between RKPB1 and RKPB2, and adjustment of the increment after RKPB2.

* Hildebrand, F. B., Introduction to Numerical Analysis, McGraw-Hill, New York, 1956, Section 6.16.5-6.

SAMPLE PROBLEM

Integrate the following system of equations:

$$DX/DT = Y$$
$$DY/DT = 4X$$
$$DZ/DT = 2Z$$

From T = 0. to T = 2. where X = 0., Y = 2. and Z = 1. at T = 0. Use an interval of integration DT = .0625 and print T, X, Y and Z for every integration step. The program necessary to accomplish this may be as follows, where symbolically in the programs: U(1) is used for X, U(2) is used for Y, and U(3) is used for Z. The derivatives of X, Y and Z are referred to as F(1), F(2) and F(3), respectively.

SAMPLE SOLUTION

```
10        EXTERNAL DERIV
20        COMMON U(3),F(3)
30        DIMENSION TEMP(16)
40        T=0.0; DT=.0625; N=3
50        U(1)=0.0; U(2)=2.0; U(3)=1.0; TF=2.0
60        PRINT 1
70      1 FORMAT(/8X,1HT,13X,1HX,13X,1HY,13X,1HZ)
80     10 CALL RKPB1(DERIV,TEMP,T,DT,U,F,N)
90        PRINT 15,T,U
100    15 FORMAT(1H 4E14.6)
110       IF (T-TF) 20,30,30
120    20 CALL RKPB2(DERIV,TEMP,T,DT,U,F,N)
130       GO TO 10
140    30 STOP
150       END
160C   DERIVATIVE EVALUATION SUBROUTINE
170       SUBROUTINE DERIV
180       COMMON U(3),F(3)
190       F(1)=U(2); F(2)=-4.*U(1); F(3)=2.*U(3)
200       RETURN
210       END

READY

*RUN *; RKPBX
```

T	X	Y	Z
0.	0.	0.200000E+01	0.100000E+01
0.625000E-01	0.124674E+00	0.198440E+01	0.113315E+01
0.125000E+00	0.247403E+00	0.193782E+01	0.128402E+01
0.187500E+00	0.366272E+00	0.186102E+01	0.145499E+01
0.250000E+00	0.479425E+00	0.175517E+01	0.164872E+01
0.312500E+00	0.585096E+00	0.162193E+01	0.186824E+01
0.375000E+00	0.681638E+00	0.146338E+01	0.211700E+01
0.437500E+00	0.767542E+00	0.128200E+01	0.239887E+01
0.500000E+00	0.841470E+00	0.108061E+01	0.271828E+01
0.562500E+00	0.902266E+00	0.862357E+00	0.308021E+01
0.625000E+00	0.948984E+00	0.630649E+00	0.349033E+01
0.687500E+00	0.980892E+00	0.389101E+00	0.395507E+01
0.750000E+00	0.997494E+00	0.141480E+00	0.448168E+01

```
 0.812500E+00   0.998531E+00 -0.108348E+00   0.507840E+01
 0.875000E+00   0.983986E+00 -0.356485E+00   0.575458E+01
 0.937500E+00   0.954086E+00 -0.599060E+00   0.652080E+01
 0.100000E+01   0.909299E+00 -0.832286E+00   0.738903E+01
 0.106250E+01   0.850322E+00 -0.105252E+01   0.837286E+01
 0.112500E+01   0.778076E+00 -0.125634E+01   0.948770E+01
 0.118750E+01   0.693688E+00 -0.144055E+01   0.107510E+02
 0.125000E+01   0.598476E+00 -0.160228E+01   0.121824E+02
 0.131250E+01   0.493925E+00 -0.173901E+01   0.138045E+02
 0.137500E+01   0.381666E+00 -0.184860E+01   0.156426E+02
 0.143750E+01   0.263451E+00 -0.192934E+01   0.177253E+02
 0.150000E+01   0.141126E+00 -0.197998E+01   0.200854E+02
 0.156250E+01   0.165982E-01 -0.199972E+01   0.227598E+02
 0.162500E+01  -0.108189E+00 -0.198826E+01   0.257902E+02
 0.168750E+01  -0.231287E+00 -0.194577E+01   0.292241E+02
 0.175000E+01  -0.350776E+00 -0.187292E+01   0.331152E+02
 0.181250E+01  -0.464792E+00 -0.177084E+01   0.375245E+02
 0.187500E+01  -0.571555E+00 -0.164113E+01   0.425208E+02
 0.193750E+01  -0.669398E+00 -0.148580E+01   0.481823E+02
 0.200000E+01  -0.756797E+00 -0.130730E+01   0.545977E+02

PRØGRAM STØP AT 140
*
```

ROMBINT

This FORTRAN program performs Romberg integration. The subroutine **ROMINT** may also be easily extracted.

INSTRUCTIONS

The program integrates the Function F(X) between the limits XO and XF, subdividing the interval enough to meet the input error criteria EPS.

The function subprogram F(X) should be typed in starting at line number 1000. For example:

```
1000 FUNCTION F(X)
1010 F = 3.4 + X**2 + COS(X)
1020 RETURN
1030 END
```

The program requests the values XO, XF, and EPS at run time.

The program outputs:

INT	=	value of the integral,
ERR	=	estimated error,
EPS	=	input error criteria,
XO, XF	=	integration limits,
NEVAL	=	number of function evaluations,
NEXTR	=	number of Romberg extrapolations.

The program will continue asking for new values of XO, XF, and EPS until an EPS ≤ 0 is input.

The subroutine version of the program may be obtained by typing

```
LIB  ROMBINT (100, 999)
```

The calling sequence for the subroutine entry is:

```
CALL ROMINT(VAL, ERR, EPS, A, B, N, MAXE)
```

where:

VAL	=	value of the integral
EPS	=	input error limit
N	=	number of function evaluations performed.
ERR	=	error estimate
A, B	=	interval ends
MAXE	=	limit on the number of extrapolations to be performed.

SAMPLE PROBLEM

Find $\int_0^1 (3.4 + X^2 + \text{Cos } X)\ dX$ with EPS = 10^{-4}

SAMPLE SOLUTION

```
*1000 FUNCTIØN F(X)
*1010 F = 3.4 + X**2 + CØS(X)
*1020 RETURN
*1030 END
*RUN

RØMBINT

XØ,XF,EPS
= 0,1,.0001
NAMELIST          ØUT
INT            0.45748041E 01
ERR            0.44703484E-06
EPS            0.10000000E-03
XØ             0.
XF             0.10000000E 01
NEVAL                     9
NEXTR                     2
XØ,XF,EPS
= 0,0,0

PRØGRAM STØP AT 29
*
```

ROOTER

This BASIC program finds the roots of a polynomial with real coefficients.

INSTRUCTIONS

Enter data in the following format:

10 DATA N, A(N), A(N-1), . . . , A(1), A(0)

where,

N is the order of the polynomial.

A(N), A(N-1), . . . , A(1), A(0) are the polynomial coefficients in descending order.

More than one data line may be used to supply coefficients for one polynomial; additional polynomials may be solved on a single run by supplying data for them on subsequent data lines--not beyond Line 299.

There are a few types of polynomials which this program is unable to solve. The program will so indicate and go on to the next case.

If the program does not converge on a root within 25 iterations, a message to that effect will be printed. The program then provides the option of continuing the iteration or stopping. If the iteration is continued, the program will print a message every 25 additional iterations until the root is found or the program is stopped. If a root is not obtained after 100 iterations, it is recommended that another method be used to solve the polynomial.

METHOD

Bairstow's method is used[1].

SAMPLE PROBLEM

Determine the roots for the following 3 polynomials:

$6 + 3X = 0$
$5 - 6X + X^2 = 0$
$4 - 7X^2 + 3X^3 = 0$

---

1 Hamming, R. W., Numerical Methods For Scientists And Engineers, McGraw Hill Book Co., New York, 1967, Pages 356-359.

SAMPLE SOLUTION

Note the following:

All 3 polynomials are solved in one run.

Each set of coefficients must be preceded by the order of the polynomial.

The coefficients must be entered in descending order.

ROOTER gives erroneous results for double imaginary roots.

```
*10 DATA 1,3,6
*11 DATA 2,1,-6,5
*12 DATA 3,3,-7,0,4
*RUN

R00TER


P0LYN0MIAL NUMBER     1 IS 0F 0RDER     1
   C0EFFICIENTS (IN DESCENDING 0RDER) ARE:
          3      6
   THE R00T IS:
        -2

P0LYN0MIAL NUMBER     2 IS 0F 0RDER     2
   C0EFFICIENTS (IN DESCENDING 0RDER) ARE:
          1     -6      5
   THE R00TS ARE:
          5  AND       1

P0LYN0MIAL NUMBER     3 IS CF 0RDER     3
   C0EFFICIENTS (IN DESCENDING 0RDER) ARE:
          3     -7      0      4
   THE R00TS ARE:
          2
          1  AND    -0.6666667

READY
*
```

SECANT

This FORTRAN subroutine uses the secant method to solve the nonlinear system of equations F(X) = 0 where F and X are N dimensional vectors.

METHOD

The vector $\hat{f}$ has as its components the N functions of N variables $f_1, f_2 \ldots f_N$. The user enters an initial guess $\hat{x}^T = (x_1, x_2, \ldots, x_N)$ from which N other vectors are determined by altering each element of $\hat{x}$ indepefidently. Each function f_i is then evaluated at each of the N+1 points. These points determine a plane for each function. In order for the method to converge each plane must intersect all others as well as the identically zero plane. The common intersection in the zero plane yields a new estimate of the solution vector. The method fails when two planes become nearly parallel as the iterative method proceeds, or if a plane becomes nearly parallel to the zero plane. This is the cause of IERR=1. A measure of the size of the vector $\hat{f}(\hat{x})$ is defined by $\| \hat{f}(\hat{x}) \|_\infty = \max_i \{ | f_i(\hat{x}) | \}$. The vector $\hat{x}_{n+1}$ obtained on the $(n+1)^{th}$ iteration is taken as the solution if $\| \hat{x}_{n+1} - \hat{x}_n \|_\infty \leq CC$. During the course of the iterative procedure the routine will remember the vector of independent variables $\hat{s}$ for which FM = $\| \hat{f}(\hat{s}) \|_\infty$ was least over all $\hat{x}$'s tried. If the routine fails for some reason (IERR$\neq$0), the user may want to re-enter with $\hat{s}$ as the initial estimate.

INSTRUCTIONS

This routine calls the routine MTINV which must be executed jointly with this program.

The calling sequence is:

CALL SECANT(N,NI,CC,FM,X,F,Q,Z,S,G,Y,IDIM,EVAL,IERR)

where,

- N is the order of the system of nonlinear equations.
- NI is the maximum number of iterations.
- CC is a convergence criterion, for example 10^{-6}.
- FM is the norm of the vector $\hat{f}(\hat{x})$ calculated internally.
- X is the one dimensional array which contains the initial guess to the solution vector for input and, if IERR=0 or 2, the solution vector for output. The initial guess vector cannot be the zero vector. The components of the initial guess vector are stored in X(1)...X(N) respectively. X must be dimensioned X(M) where M.GE.N+1. (X corresponds to the $\hat{x}$ vector.)
- F is the one dimensional array which contains the N functions evaluated for the X vector. F must be dimensioned F(M) where M.GE.N+1. (F corresponds to the $\hat{f}$ vector.
- Q is a one dimensional array used internally. Q must be dimensioned Q(M), where M.GE.N+1.

- Z is a one dimensional array which contains the latest estimate of the solution vector if IERR=1. Z must be dimensioned Z(M) where M.GE.N+1.

- S is a one dimensional array used internally. S must be dimensioned S(M), where M.GE.N+1.

- G is a two dimensional array used internally. G must be dimensioned G(IDIM,M), where M.GE.N+1.

- Y is a two dimensional array used internally. Y must be dimensioned Y(IDIM,M), where M.GE.N+1.

- IDIM is the first dimension of the G and Y arrays. IDIM.GE.N+1.

- EVAL is a subroutine supplied by the user to evaluate $\hat{f}(\hat{x})$. It is of the form:

  SUBROUTINE EVAL(F,X)

  where F and X are as defined above.

- IERR is returned as follows:

IERR=0.	Normal return X contains the solution vector. F contains the function vector.
IERR=1	Matrix inversion has deteriorated. Z contains the solution vector. F contains the function vector.
IERR=2	Maximum number of iterations has been exceeded. X contains the solution vector. F contains the function vector.

SAMPLE PROBLEM

Solve the system:

$$F_1(X) = -2 + X_1 + X_2$$

$$F_2(X) = 1 - X_1X_2$$

starting with an initial guess of (1,-1). The solution is (1,1).

SAMPLE SOLUTION

```
010 DIMENSIØN X(3),F(3),Q(3),Z(3),S(3),G(3,3),Y(3,3)
020 EXTERNAL EVAL
030 X(1)=1.
040 X(2)=-1.
050 N=2
060 NI=10
070 CC=1.E-6
080 CALL SECANT(N,NI,CC,FM,X,F,Q,Z,S,G,Y,3,EVAL,IERR)
090 IF(IEFF.EQ.1)GØ TØ 100
100 IF(IERR.EQ.2)GØ TØ 200
110 PRINT:"NØRMAL RETURN"
120 PRINT 600
130 PRINT:(X(I),I=1,N)
140 PRINT 700
150 PRINT:(F(I),I=1,N)
160 GØ TØ 800
170 100 PRINT:"MATRIX INVERSIØN HAD DETERIØRATED"
180 PRINT 600
190 PRINT:(Z(I),I=1,N)
200 PRINT 700
210 PRINT:(F(I),I=1,N)
220 GØ TØ 800
230 200 PRINT:"MAXIMUM NUMBER ØF ITERATIØNS EXCEEDED"
240 PRINT 600
250 PRINT:(X(I),I=1,N)
260 PRINT 700
270 PRINT:(F(I),I=1,N)
280 600 FØRMAT(//" SØLUTIØN VECTØR"//)
290 700 FØRMAT(//" FUNCTIØN VECTØR"//)
300 800 STØP;END
310 SUBRØUTINE EVAL(F,X)
320 DIMENSIØN X(3),F(3)
330 F(1)=-2.+(X(1)+X(2))
340 F(2)=1.-X(1)*X(2)
350 RETURN
360 END

READY

*RUN *;SECANT;MTINV
NØRMAL RETURN


SØLUTIØN VECTØR

    9.9999975E-01    1.0000002E+00


FUNCTIØN VECTØR

   -2.2351742E-08    2.2351797E-08

PRØGRAM STØP AT 300
*
```

SIMEQN

This BASIC program solves sets of simultaneous linear equations with N unknowns.

NOTE:

The algorithm used may result in excessive running time for large problems.

INSTRUCTIONS

To use this program enter coefficients in the equations in data statements, starting in statement 11 with the first coefficient of the first equation, and ending with the N TH coefficient of the N-TH equation. All zero coefficients must be entered in their proper place. The right-side constant terms of the equations are then entered in subsequent data statements. If additional cases with the same coefficient matrix but different right sides are to be run, they may all be run at once by simply entering additional data statements with the right-side values of the other cases. A data statement at line 10 [preceding all the above] is used to specify the number of systems to be solved, and the number of equations [and hence variables] in the system. Thus, the two systems:

```
3X + 5Y - 2Z     =  9          3X + 5Y - 2Z     = 19
7X +  Y          = -3          7X +  Y          + -3
 X - 7Y + 9Z     = 14           X - 7Y + 9Z     =  8
```

Could be solved by typing:

```
10 DATA 2,3
11 DATA 3,5,-2,7,1,0,1,-7,9
12 DATA 9,-3,14,19,-3,8
RUN
```

Solutions are proofed by multiplying the vector by the original coefficient matrix.

Additional instructions may be found in the listing.

SAMPLE PROBLEM

Solve the following sets of simultaneous equations:

```
25W    -2.7X    +38.7Y    +  Z    =    115.76    10
         18X    - 5.8Y    - 3Z    =     11.5     20
 6W    +  9X    +  18Y             =     55       30
11W              +  87Y    +41Z    =     17       40
```

The extreme right-hand column of numbers is a second set of right-hand values.

To use this program to solve the two sets, prepare the following data tape:

```
10 DATA 2,4
11 DATA 25, -2.7,38.7, 1,0, 18, -5.8, -3
12 DATA 6, 9, 18, 0, 11, 0, 87, 41
13 DATA 115.76, 11.5, 55, 17, 10, 20, 30, 40
```

In line 10, the 2 means two sets of equations and the 4 means each set consists of four equations. In the 11 and 12 lines, the 0's are necessary to represent any missing (zero coefficient) terms in the set. Line 13 contains the right-hand values for both sets.

NOTE:

The answers for the first set would be read thus:

```
W = 1.25315        Y = 2.34878
X =  .578122       Z =-4.90557
```

The proof lines merely use the calculated values to evaluate the left-hand side of each equation to prove that they give the right-hand value.

SAMPLE SOLUTION

```
*10 DATA 2, 4
*11 DATA 25, -2.7, 38.7, 1, 0, 18, -5.8, -3
*12 DATA 6, 9, 18, 0, 11, 0, 87, 41
*13 DATA 115.76, 11.5, 55, 17, 10, 20, 30, 40
*RUN

SIMEQN

SØLUTIØN FØR LINEAR SYSTEM ØF ØRDER     4

                    INDEX:

          1                2                3                4

SØLUTIØN VECTØR FØR CASE     1

     1.253154         0.5781224         2.348776        -4.905567
PRØØF ØF SØLUTIØN FØR CASE     1

      115.76             11.5               55               17

SØLUTIØN VECTØR FØR CASE     2

    -1.951556         1.308704         1.662833        -2.029253
PRØØF ØF SØLUTIØN FØR CASE     2

     9.999999               20               30               40

READY
*
```

SOLN

This FORTRAN function finds a zero of an arbitrary function.

INSTRUCTIONS

The calling sequence for the entry SOLN is:

X = SOLN(IND,FUNC,BOT,TOP,EPS,IERR)

where,

- X is the zero

  IND = 1 the convergence test is absolute.
  IND = 2 the convergence test is relative.

- FUNC is the name of the user defined function subprogram of the form Y = FUNC(X). FUNC must be defined by an external statement as shown in the Sample Problem.

- BOT is the lower bound of the independent variable.

- TOP is the upper bound of the independent variable.

- EPS is the convergence criterion.

- IERR is an error indicator as follows:

  IERR = 0 Satisfied criterion for type of convergence desired.

  IERR = 1 Could not bracket a root before mesh size became too small.

  IERR = 2 A root was bracketed but the maximum number of iterations was exceeded before the absolute convergence criterion was satisfied.

  IERR = 3
  1. A root was bracketed.
  2. Maximum number of iterations was exceeded.
  3. Relative convergence criterion could not be applied.
  4. Absolute convergence criterion was satisfied.

  IERR = 4
  1. A root was bracketed.
  2. Maximum number of iterations was exceeded.
  3. Relative convergence criterion could not be applied.
  4. Absolute convergence criterion was not satisfied.

  IERR = -1 EPS not positive.

  IERR = -3
  1. A root was bracketed.
  2. Maximum number of iterations was exceeded.
  3. Relative convergence criterion could not be applied.
  4. Absolute convergence criterion was satisfied.

IERR = -4

1. A root was bracketed.
2. Maximum number of iterations was exceeded.
3. Neither convergence criterion was satisfied.

RESTRICTION

Maximum number of iterations is 100.

METHOD

The zero is bracketed, i.e., function positive on the side, negative on other, whereupon the interval having technique is used.

If possible, TOP and BOT should bracket the zero as indicated above to avoid the search for bracketing values. If no zero can be detected between BOT and TOP, the routine FUNC is evaluated either 100 times, or (TOP-BOT)/EPS times--whichever is less.

SAMPLE PROBLEM

Find the zero of the function $X^2 - 4$ between the limits 0. and 5.0. Since the zero (+2.0) is between these limits, use an absolute convergence test of EPS = 0.001.

SAMPLE SOLUTION

```
010         EXTERNAL FUNC
020         IERR=0
030         IND=1; FPS=.001
040         X=SØLN(IND,FUNC,0.,5.,FPS,IERR)
050         IF(IERR) 1,2,1
060       1 PRINT: " ERRØR NUMBER",IERR
070         STØP
080       2 PRINT: " THE FUNCTIØN HAS A ZERØ AT X =",X
090         STØP; END
100         FUNCTIØN FUNC(X)
110         FUNC=X*X-4.
120         RETURN; END

READY

*RUN *;SØLN
 THE FUNCTIØN HAS A ZERØ AT X =    2.0002747E+00

PRØGRAM STØP AT 90
*
```

SPEIG1

This FORTRAN subroutine finds the eigenvalues and eigenvectors of a real non-symmetric matrix which can be expressed as a product of two symmetric matrices, one of which must be positive definite.

INSTRUCTIONS

The calling sequence is:

CALL SPEIG1 (IND,A,B,N,TEMP1,TEMP2,C)

where,

- IND determines which of four special eigenproblems is to be solved for the eigenvector matrix X, and the diagonal eigenvalue matrix J:

IND = 1	The routine solves ABX=XJ
IND = 2	The routine solves BAX=XJ
IND = 3	The routine solves AX=BXJ
IND = 4	The routine solves BX=AXJ

- A is the name of the positive definite matrix, i.e., the eigenvalues of A must be positive. If A is not positive definite, the routine sets IND to -1 and returns control to the calling program.

- B is the name of the real symmetric matrix which may or may not be positive definite.

- N is the order of the A and B matrices.

- TEMP1 and TEMP2 are arrays used for internal storage, each containing at least N locations.

- C is the name of a double dimension array used by the routine for storage purposes. Dimensions are the same as for A and B.

RESTRICTIONS

The order of the matrices must not exceed 15.

The first dimension of A, B, and C must be 15.

The subroutine EIG1 must be included in the RUN statement.

METHOD

The diagonal matrix J is stored in the diagonal elements of the A matrix, thus destroying the original positive definite matrix. The eigenvector matrix X is stored columnwise in B, thus destroying the original real symmetric matrix.

SAMPLE PROBLEM

Find the eigenvalues and eigenvectors of the matrix C which is the product of the real symmetric matrices A and B, in that order (IND = 1).

A and B are defined as:

$$A = \begin{pmatrix} 4. & 2. & 1. \\ 2. & 4. & 2. \\ 1. & 2. & 4. \end{pmatrix} \qquad B = \begin{pmatrix} 1. & -1. & -1. \\ -1. & 2. & 4. \\ -1. & 4. & 6. \end{pmatrix}$$

SAMPLE SOLUTION

```
010       DIMENSIØN A(15,3),B(15,3),U(15,3),TEMP1(3),TEMP2(3)
020       A(1,1)=4.0; A(2,2)=4.0; A(3,3)=4.0; B(3,2)=4.0
030       B(2,3)=4.0; A(1,2)=2.0; A(2,1)=2.0; A(3,2)=2.0
040       B(2,2)=2.0; A(3,1)=1.0; A(1,3)=1.0; B(1,1)=1.0
050       B(2,1)=-1.0; B(3,1)=-1.0; B(1,2)=-1.0; B(1,3)=-1.0
060       A(2,3)=2.0; B(3,3)=6.0
070       CALL SPEIG1(1,A,B,3,TEMP1,TEMP2,U)
080       DØ 10 I=1,3
090       PRINT 20,I,A(I,I)
100    10 PRINT 30,I,(B(J,I),J=1,3)
110    20 FØRMAT(/11H EIGENVALUE,I2,1X,E20.7)
120    30 FØRMAT(7H VECTØR,I2,1X,3E15.6)
130       STØP
140       END

READY

*RUN *;SPEIG1;EIG1

EIGENVALUE 1          0.2497657E+01
VECTØR 1     0.180873E+01    0.174069E+00    0.201258E+00

EIGENVALUE 2          0.4478955E+02
VECTØR 2     0.566109E+00    0.150982E+01    0.187504E+01

EIGENVALUE 3         -0.1287220E+01
VECTØR 3    -0.638768E+00   -0.130006E+01    0.666124E+00

PRØGRAM STØP AT 130
*
```

STIRLING

This FORTRAN program consists of a subroutine and a driver program that calculates factorials of positive integers using Stirling's approximation. The subroutine FCTRL may also be easily extracted.

INSTRUCTIONS

To use the freestanding program type RUN. When requested, enter the integer whose factorial is desired. If the integer is less than or equal to 30, the program will print the factorial correct to eight digits. If the integer is greater than 30, the program will give the upper and lower bounds for the factorial. The program will continue requesting more integers until a negative integer is entered.

To use the subroutine, delete the driver coding in lines 1-99. The calling sequence is:

```
CALL FCTRL (N, OUT, LEFT, XLOG)
```

where:

```
XLOG = LOG10 (N!)
  N! = OUT * 10 ** LEFT
```

If N is less than or equal to 30, then LEFT = 0 and OUT is the approximation to N! in standard floating-point form. If N is greater than 30, then OUT is the mantissa in floating-point form 0.XXXXXXXXEO and LEFT is the exponent.

METHOD

Stirling's approximation for N! is

$$\sqrt{2\pi N}\ (N/e)^N\ (12N/(12N-1)) > N! > \sqrt{2N\pi}\ (N/e)^N$$

The subroutine FCTRL approximates N! using the lower bound of Stirling's approximation.

RESTRICTIONS

N must be a nonnegative integer less than 13020810.

SAMPLE SOLUTION

```
*RUN
THIS PROGRAM USES STIRLINGS APPROXIMATION TO CALCULATE THE
FACTORIAL OF N.  ENTER N<1 TO STOP PROGRAM.
ENTER N
= 1
N!=1.0000000E+00
ENTER N
= 30
N!=2.6525286E+32
ENTER N
= 31
8.2228645E+33 >N!> 8.2007601E+33
ENTER N
= 50
3.0414086E+64 >N!> 3.0363396E+64
ENTER N
= 70
1.1978567E+100 >N!> 1.1964307E+100
ENTER N
= 90
1.4857098E+138 >N!> 1.4843341E+138
ENTER N
= 100
9.3326029E+157 >N!> 9.3248258E+157
ENTER N
= 1000
4.0236475E+2567 >N!> 4.0233122E+2567
ENTER N
= 5789
7.2666520E+19269 >N!> 7.2665475E+19269
ENTER N
= 130202809
1.0000000E+1000000000 >N!> 1.0000000E+1000000000
ENTER N
= 130202810
N TOO LARGE, ONLY XLOG MEANINGFUL. (N=  130202810)
N!=0.
ENTER N
= -1

PROGRAM STOP AT 28
*
```

SYMEIG

This FORTRAN program calculates the eigenvectors and eigenvalues of a real, symmetrical matrix.

METHOD

The method used is Jacobi's Iteration Method, which was adapted for computer use by Von Neuman. The method consists of applying to the matrix a system of plane rotations given by orthogonal matrices designed to reduce the off-diagonal elements to zero. The eigenvalues are then the diagonal elements of the original matrix and, if the eigenvectors were desired, they are developed as the columns of the product of the orthogonal matrices.

INSTRUCTIONS

Enter data as requested. For further instructions run the program.

SAMPLE SOLUTION

```
SYMEIG


*RUN

DØ YØU WANT INSTRUCTIØNS?
= YES
INSTRUCTIØNS FØR SYMEIG
THIS PRØGRAM CALCULATES THE EIGENVECTØRS AND EIGENVALUES ØF A
REAL, SYMMETRICAL MATRIX.  THE METHØD USED IS JACØBI'S LITERATIØN
METHØD. THE METHØD CØNSISTS ØF APPPLYING TØ THE MATRIX A SYSTEM
ØF PLANE RØTATIØNS GIVEN BY ØRTHØGØNAL MATRICES MADE TØ REDUCE
THE ØFF-DIAGØNAL ELEMENTS TØ ZERØ.  THIS PRØGRAM USES FØUR
ARGUMENTS.  A IS THE NAME ØF A TWØ-DIMENSIØNAL ARRAY CØNTAINING
THE REAL, SYMMETRIC MATRIC IN ITS FIRST N RØWS AND CØLUMNS
R IS THE NAME ØF THE TWØ-DIMENSIØNAL ARRAY WHICH WILL CØNTAIN
THE EIGENVECTØRS IN ITS FIRST N CØLUMNS.
N IS AN INTEGER VARIABLE ØR CØNSTANT GIVING THE ØRDER
ØF THE MATRIX.
MV IS AN INTEGER VARIABLE ØR CØNSTANT WHICH MUST BE 0 ØR 1.
IF IT IS 0 BØTH EIGENVECTØRS AND EIGENVALUES ARE FØRMED.
IF IT IS ØNE ØNLY THE EIGENVALUES ARE FØUND.
ENTER THE ØRDER ØF MATRIX AND THE MATRIX SEPARATED BY CØMMAS.
= 3
= 1,1,.5
= 1,1,.25
= .5,.25,2
THE MATRIX IS
    1.0000000E+00    1.0000000E+00    5.0000000E-01
    1.0000000E+00    1.0000000E+00    2.5000000E-01
    5.0000000E-01    2.5000000E-01    2.0000000E+00
EACH EIGENVALUE FØLLØWED BY CØRRESPØNDING EIGENVECTØR
   -1.6647290E-02
   -7.2120713E-01    6.8634924E-01    9.3727956E-02
    1.4801212E+00
    4.4428103E-01    5.6210938E-01   -6.9760113E-01
    2.5365255E+00
    5.3148334E-01    4.6147330E-01    7.1032929E-01

PRØGRAM STØP AT 85
*
```

TMFCEV

This BASIC program evaluates time functions which are sums of exponentials and exponential sine-cosine terms.

INSTRUCTIONS

To use this program enter input data in the following format:

```
20    DATA  NP,N1,N2,TO,DELTA-T,SIGMA
```

where:

NP=TOTAL NUMBER OF POINTS TO BE COMPUTED
N1=NUMBER OF EXPONENTIAL TERMS
N2=NUMBER OF SINE-COSINE EXPONENTIAL TERMS
TO=TIME OF FIRST POINT
DELTA-T=TIME BETWEEN POINTS
SIGMA=STANDARD DEVIATION OF THE NOISE

NOTE:

If additive noise is not desired, SIGMA = 0.

Parameters of the function are entered as follows:

```
41      DATA  C[1],C[2],C[3],...,S[1],S[2],S[3],...
60      DATA  A[1],A[2],A[3],...,B[1],B[2],B[3],...
79      DATA  W[1],W[2],W[3],...,G[1],G[2],G[3],...
```

Only statement numbers between 41 and 79 inclusive may be used.

Maximum number of points permissible is 500 and 2N1+4N2 = 20.

The computation of E↑-X, where X is large may result in excessive running time.

Additional instructions may be found in the listing.

SAMPLE PROBLEM

Evaluate a time function of this form:

```
C*EXP(-S*T)
```

C	S
-0.0178189	1.5662
0.0166119	16.2565
0.00450666	136.889

```
FORM:        (A*COS(W*T)+B*SIN(W*T))*EXP(-G*T)
```

A	B	W	G
15.438	-.372211	491.834	200.474

Type the following data:

```
20    DATA   30, 3, 1, 0, .001, 0
50    DATA   -1.78189E-2, 1.66119E-2, 4.50666E-3, 1.5662, 16.2565, 136.889
60    DATA   15.438, -.37221, 491.834, 200.474
```

NOTE:

In the data entered on line 20, 30 is the total number of points desired, 3 is the number of exponential terms, 1 is the number of sine-cosine exponential terms, 0 is the time of the first point, .001 is the time between the points and 0 is the standard deviation of the noise.

The parameters of the functions, which are on lines 50 and 60, can be entered from lines 41-79.

TMFCEV provides for adding random numbers from a uniform distribution (-1/2, 1/2) with a standard deviation σ (standard deviation of the noise). If additive noise is not desired, input σ as zero.

SAMPLE SOLUTION

```
*20 DATA 30,3,1,0,.001,0
*50 DATA -1.78189E-2,1.66119E-2,4.50666E-3,1.5662,16.2565,136.889
*60 DATA 15.438,-.37221,491.834,200.474
*RUN


NØISE SIGMA =     0

TERMS ØF FØRM    C*EXP(-S*T)    ARE:

                        C                S

                  -0.0178189          1.5662
                   0.0166119         16.2565
                   0.0045067         136.889

TERMS ØF FØRM    (A*CØS(W*T)+B*SIN(W*T))*EXP(-G*T)    ARE:

                        A                B               W               G

                    15.438         -0.37221         491.834         200.474

FIRST PØINT AT T1,SPACING=T2
T1=            0    T2=    0.001

DATA PØINTS ARE:

     15.4413         10.99474         5.521522        0.6030367       -2.827604
   -4.483233        -4.574454        -3.597412        -2.126857       -0.6597066
    0.4720381        1.122184         1.302155         1.126335        0.7520941
    0.3298639       -0.028716        -0.263328        -0.3617165      -0.3465941
   -0.2589358       -0.1426456       -0.0336229        0.0457423       0.0871895
    0.0937954        0.0755308        0.0447195        0.0124639      -0.013472

READY
*
```

TNT1

This is a FORTRAN function intended to do a single variable table look-up and Lagrangian interpolation of specified order.

INSTRUCTIONS

The calling sequence is:

```
Y = TNT1(X,NTAB,XTAB,YTAB,NPT,IERR)
```

where,

- Y is the interpolated value.
- X is the value of the independent argument.
- NTAB is the number of elements in the table.
- XTAB is the name of the independent variable table.
- YTAB is the name of the dependent variable table.
- NPT is the number of points over which the interpolation is performed.
- IERR is an error return as follows:

	IERR
$X \leq XTAB(1)$	-1
$XTAB(1) \leq X \leq XTAB(NTAB)$	0
$X > XTAB(NTAB)$	1

RESTRICTIONS

The elements of the independent variable table must be in monotonic ascending order.

The library subprogram TLU1 must be used with this subprogram, as shown in the Sample Problem.

METHOD

The order of interpolation is N = MIN(NPT-1,NTAB-1). The best N+1 points are selected for the interpolation.

Special cases are:

N = 0	No interpolation
N = 1	Linear interpolation
N = 2	Parabolic interpolation

When the argument is outside the range of the independent variable table, TNT1 is set to 0.

SAMPLE PROBLEM

Find the interpolated value in the YTAB table for a value of 5.1 in the XTAB table. Compute the interpolated value by both linear and parabolic interpolation. The tables are defined as follows:

YTAB	XTAB
0.0	0.0
10.	1.0
20.	2.0
30.	3.0
40.	4.0
50.	5.0
60.	6.0
70.	7.0
80.	8.0
90.	9.0

SAMPLE SOLUTION

```
10      DIMENSIØN XTAB(10),YTAB(10)
20      XTAB(1)=0.0; YTAB(1)=0.0
30      DØ 10 I=2,10
40      XTAB(I)=XTAB(I-1)+1.0
50   10 YTAB(I)=YTAB(I-1)+10.
60      X=5.1
70      NTAB=10
80      NPT=2
90      IERR=0
100     Y=TNTI(X,NTAB,XTAB,YTAB,NPT,IERR)
110     IF (IERR) 20,30,20
120  20 PRINT 21
130  21 FØRMAT(/22H ARGUMENT NØT IN TABLE)
140     IERR=0
150     GØ TØ 40
160  30 PRINT 31,Y
170  31 FØRMAT(/26H LINEAR INTERPØLATED VALUE,1X,E20.7)
180  40 NPT=3
190     Y=TNTI(X,NTAB,XTAB,YTAB,NPT,IERR)
200     IF (IERR) 50,60,50
210  50 PRINT 21
220     GØ TØ 70
230  60 PRINT 61,Y
240  61 FØRMAT(/29H PARABØLIC INTERPØLATED VALUE,1X,E20.7)
250  70 STØP
260     END
```

```
READY
*RUN *,TNT1,TLU1
LINEAR INTERPØLATED VALUE          0.5100000E+02
PARABØLIC INTERPØLATED VALUE       0.5100000E+02
PRØGRAM STØP AT 250
*
```

TNT2

This FORTRAN function is intended to do a double variable table look-up and linear interpolation.

INSTRUCTIONS

The calling sequence is:

Y = TNT2(X1,X2,NTAB1,NTAB2,XTAB1,XTAB2,YTAB,IERR1,IERR2,IDIM)

where,

- Y is the interpolated value.
- X1 is the value of the first independent argument.
- X2 is the value of the second independent argument.
- NTAB1 is the number of elements in the first independent variable table.
- NTAB2 is the number of elements in the second independent variable table.
- XTAB1 is the name of the first independent variable table.
- XTAB2 is the name of the second independent variable table.
- YTAB is the name of the dependent variable table.
- IERR1 is an error return as follows:

	IERR1
X1≤XTAB(1)	-1
XTAB1(1)≤X1≤XTAB1(NTAB1)	0
X1>XTAB1(NTAB1)	1

- IERR2 is an error return as follows:

	IERR2
X2≤XTAB2(1)	-1
XTAB2(1)≤X2≤XTAB2(NTAB2)	0
X2>XTAB2(NTAB2)	1

- IDIM is the number of elements in the first dimension of the dependent variable.

RESTRICTIONS

The elements of each independent variable table must be in monotonic ascending order.

The library subprogram TLU1 must be used with this subprogram, as shown in the Sample Problem.

METHOD

The table is in the following format:

	YTAB2(1)	. . .	XTAB2(NTAB2)
XTAB1(1)	YTAB(1,1)	. . .	YTAB(1,NTAB2)
.	.	. . .	.
.	.	. . .	.
.	.	. . .	.
.	.	. . .	.
XTAB1(NTAB1)	YTAB(NTAB1,1)	. . .	YTAB(NTAB1,NTAB2)

When either argument is outside the range of its independent variable table, TNT2 is set to 0.

SAMPLE PROBLEM

Perform a table look-up and linear interpolation for X1=2.5 and X2=15.0. The tables are described below.

		XTAB2							
		-50.	-10.	0.0	10.	20.	30.	50.	
X	1.0	1.0	1.0	1.0	2.0	3.0	4.0	4.0	YTAB ROW 1
T									
A	2.0	1.0	1.0	2.0	3.5	5.0	6.0	6.0	YTAB ROW 2
B									
1	3.0	1.0	1.8	2.0	2.2	2.4	2.6	3.0	YTAB ROW 3
	4.0	0.0	4.0	5.0	6.0	7.0	8.0	10.0	YTAB ROW 4

SAMPLE SOLUTION

```
010       DIMENSIØN XTAB1(4),XTAB2(7),YTAB(50,7)
020       XTAB1(1)=1.0; YTAB(1,1)=1.0; YTAB(1,2)=1.0
030       YTAB(1,3)=1.0; YTAB(2,1)=1.0; YTAB(2,2)=1.0
040       YTAB(3,1)=1.0
050       XTAB1(2)=2.0; YTAB(1,4)=2.0; YTAB(2,3)=2.0
060       YTAB(3,3)=2.0
070       XTAB1(3)=3.0; YTAB(1,5)=3.0; YTAB(3,7)=3.0
080       XTAB1(4)=4.0; YTAB(4,2)=4.0; YTAB(1,6)=4.0
090       YTAB(1,7)=4.0
100       YTAB(3,2)=1.8
110       YTAB(4,1)=0.0; XTAB2(3)=0.0
120       YTAB(4,3)=5.0; YTAB(2,5)=5.0
130       YTAB(3,4)=2.2
140       YTAB(4,4)=6.0; YTAB(2,6)=6.0
150       YTAB(2,7)=6.
160       YTAB(3,5)=2.4
170       YTAB(4,5)=7.0
180       YTAB(3,6)=2.6
190       YTAB(4,6)=8.0
200       YTAB(4,7)=10.; XTAB2(4)=10.
210       XTAB2(1)=-50.
220       XTAB2(2)=-10.
230       XTAB2(5)=20.
240       XTAB2(6)=30.
250       XTAB2(7)=50.
260       YTAB(2,4)=3.5
270       IERR1=0; IERR2=0
280       X1=2.5
290       X2=15.
300       Y=TNT2(X1,X2,4,7,XTAB1,XTAB2,YTAB,IERR1,IERR2,50)
310       IF (IERR1) 20,10,20
320    10 IF (IERR2) 20,11,20
330    11 PRINT 12,Y
340    12 FØRMAT(/" TNT2    INTERPØLATED VALUE",E20.7)
350       GØ TØ 30
360    20 PRINT 21,X1,X2
370    21 FØRMAT(/9H ARGUMENT,F10.3,3H ØR,F10.3,13H NØT IN TABLE)
380    30 STØP
390       END

READY

*RUN *;TNT2;TLU1

TNT2    INTERPØLATED VALUE        0.3275000E+01

PRØGRAM STØP AT 380
```

TNT2A

This FORTRAN function is intended to do a double variable table look-up and linear interpolation with a single arrayed table. For purposes of discussion, the first independent variable table will be assumed to be vertical, and the second horizontal.

INSTRUCTIONS

The calling sequence is:

Y = TNT2A(X1,X2,NTAB1,NTAB2,XTAB1,XTAB2,YTAB,IERR1,IERR2)

where,

- Y is the interpolated value.
- X1 is the value of the first independent argument.
- X2 is the value of the second independent argument.
- NTAB1 is the number of elements in the first independent variable table.
- NTAB2 is an array containing the number of elements in each row of the second independent variable table.
- XTAB1 is the name of the first independent variable table.
- XTAB2 is the name of the second independent variable table. Elements are stored sequentially by rows.
- YTAB is the name of the dependent variable table. Elements are stored sequentially by rows.
- IERR1 is an error return as follows:

	IERR1
X1<XTAB(1)	-
XTAB1(1)≤X1≤XTAB1(NTAB1)	0
X1>XTAB1(NTAB1)	+

- IERR2 is an error return as follows:

	IERR2
X2<XTAB2(1)	-
XTAB2(1)≤X2≤XTAB2(NTAB2)	0
X2>XTAB(NTAB2)	+

RESTRICTIONS

The elements of the first independent variable table must be in monotonic ascending order.

The elements of each row of the second independent variable table must be in monotonic ascending order.

The range of values in the second independent variable table must overlap in adjacent rows.

There must be at least two rows and two elements in each row.

The library subroutine programs TLU1 and TNT1 must be used with this subroutine. (See Sample Solution.)

METHOD

When either argument is outside the range of its independent variable table, TNT2A is set to 0. The table format is best illustrated with the following example.

```
XTAB1
  1.      0.    10.    20.    30.
            1.     2.     3.     4.

  2.    -10.     0.    20.    30.
            1.     2.     5.     6.

  3.    -50.     0.    50.
            1.     2.     3.

  4.    -50.    50.
            0.    10.
```

The tables are as follows:

```
NTAB1=4
NTAB2=4,4,3,2
XTAB1=1.,2.,3.,4.
XTAB2=0.,10.,20.,30.,-10.,0.,20.,30.,-50.,0.,50.,-50.,50.
YTAB=1.,2.,3.,4.,1.,2.,5.,6.,1.,2.,3.,0.,10.
```

SAMPLE PROBLEM

Perform a linear table look-up for X1=2. 5 and X2=15. 0. Use the table described in the method section.

SAMPLE SOLUTION

```
010       DIMENSI0N NTAB2(4),XTAB1(4),XTAB2(13),YTAB(13)
020       NTAB2(1)=4; NTAB2(2)=4; NTAB1=4
030       NTAB2(3)=3; NTAB2(4)=2
040       XTAB1(1)=1.0; YTAB(1)=1.0; YTAB(5)=1.0; YTAB(9)=1.0
050       XTAB1(2)=2.0; YTAB(2)=2.0; YTAB(6)=2.0; YTAB(10)=2.0
060       XTAB1(3)=3.0; YTAB(3)=3.0; YTAB(11)=3.0
070       XTAB1(4)=4.0; YTAB(4)=4.0
080       XTAB2(1)=0.0; XTAB2(6)=0.0; XTAB2(10)=0.0
090       YTAB(12)=0.0
100       XTAB2(2)=10.0; YTAB(13)=10.0
110       XTAB2(3)=20.0; XTAB2(7)=20.0
120       XTAB2(4)=30.0; XTAB2(8)=30.0
130       XTAB2(5)=-10.0
140       XTAB2(9)=-50.0; XTAB2(12)=-50.0
150       XTAB2(11)=50.0; XTAB2(13)=50.0
160       YTAB(7)=5.0
170       YTAB(8)=6.0
180       IERR1=0; IERR2=0
190       X1=2.5; X2=15.0
200       Y=TNT2A(X1,X2,NTAB1,NTAB2,XTAB1,XTAB2,YTAB,IERR1,IERR2)
210       IF (IERR1) 20,10,20
220    10 IF (IERR2) 20,11,20
230    11 PRINT 12,Y
240    12 F0RMAT(/" TNT2A    INTERP0LATED VALUE",E20.7)
250       G0 T0 30
260    20 PRINT 21, X1,X2
270    21 F0RMAT(/9H ARGUMENT,F10.3,3H 0R,F10.3,13H N0T IN TABLE)
280    30 ST0P
290       END

READY

*RUN *;TNT2A;TNT1;TLU1

TNT2A    INTERP0LATED VALUE        0.3275000E+01

PR0GRAM ST0P AT 280
*
```

ZCOP

This FORTRAN program approximates the roots of a polynomial of the form

$$P(z) = \sum_{n=0}^{I} A_n Z^n$$

where A_n is complex.

INSTRUCTIONS

Instructions for the format of the input data are generated by the program, as shown in the Sample Problem. The program permits input errors to be corrected and generates the instructions required to make the corrections.

After the zeros of the polynomial have been approximated, the program reconstructs the polynomial coefficients using these approximations. Both the zeros and the reconstructed polynomial coefficients are printed by the program.

After the solution for the first polynomial has been computed, the program permits the user to define a new polynomial or discontinue the execution process.

RESTRICTIONS

The degree of the polynomial must not exceed 25.

The imaginary part of a root is set to zero if its magnitude is less than or equal to .0001 times the magnitude of the real part.

METHOD

The method used to approximate the zeros of the polynomial is a modified Downhill-Newton method[1].

This program uses the subroutines ZCOP2 (see sample solution).

SAMPLE PROBLEM

Find the roots of the following complex polynomial:

$$(1+I)X^3 + (-5-3I)X^2 + (18-6I)X + 0$$

---

* Lilley, F. E., Zeroes Of A Polynomial, General Electric Company, Technical Information Series Publication, 65SD531.

SAMPLE SOLUTION

```
*RUN ZC0P;ZC0P2
D0 YOU DESIRE USER INSTRUCTI0NS, TYPE YES 0R N0
= YES
THIS PR0GRAM FINDS THE ZER0S 0F A P0LYN0MIAL 0F THE F0RM:

A(1)+AI(1)+(A(2)+AI(2))*Z+(A(3)+AI(3))*(Z**2)+..+(A(N+1)+AI(N+1))*(Z**N)

WHERE A(I) IS THE REAL PART 0F THE C0MPLEX C0EFFICIENT AND AI(I) IS THE
IMAGINARY PART 0F THE C0MPLEX C0EFFICIENT. THE DEGREE N CANN0T
EXCEED 25. INPUT F0RMAT IS FREE FIELD: A(I) AND AI(I), ARE REAL AND
N IS INTEGER. C0EFFICIENTS ARE TYPED IN L0W 0RDER FIRST.
TYPE A(1),AI(1),A(2),AI(2)ETC.

N0W Y0U TRY IT

DEGREE
= 3
C0EFFICIENTS
= 0,0  18,-6  -5,-3  1,1
ANY C0RRECTI0NS, TYPE YES 0R N0
= N0
R00T N0.          REAL PART            C0MPLEX PART
  1           0.                     0.
  2           0.40000000E+01         0.19999999E+01
  3           0.                    -0.29999999E+01

                    REC0NSTRUCTED C0EFFICIENTS
SUBSCRIPT         REAL PART            C0MPLEX PART
  1           0.                     0.
  2           0.17999999E+02        -0.60000001E+01
  3          -0.50000000E+01        -0.30000000E+01
  4           0.10000000E+01         0.10000000E+01

D0 YOU WISH TO S0LVE AN0THER P0LYN0MIAL, TYPE YES 0R N0
= N0

PR0GRAM STOP AT 0
*
```

ZCOP2

This FORTRAN subprogram program finds the roots of a polynomial from its complex coefficients.

INSTRUCTIONS

The calling sequence for this subprogram is:

CALL CDOWNH(A, AI, N, RR, CR)

where,

- A is the real part of the coefficient array.
- AI is the imaginary part of the coefficient array.
- N is the degree of the polynomial.
- RR is the array of the real parts of the roots.
- CR is the array of the imaginary parts of the roots.

The polynomial coefficients must be stored in ascending powers of the variable, i.e., constant term first.

RESTRICTIONS

The degree, N, must not exceed 25.

The imaginary part of a root is set to zero if its magnitude is less than or equal to .0001 times the magnitude of the real part.

METHOD

Modified Downhill-Newton scheme.

The ceofficients are reconstructed from the computed roots and stored in A, thus destroying the original polynomial coefficients[1].

SAMPLE PROBLEM

Find the roots of the following complex polynomial:

$$(1 + I)\ X^3 + (-5 - 3I)\ X^2 + (18 - 6I)\ X + 0$$

[1] Lilley, F.E., Zeroes of a Polynomial, General Electric Company, Technical Information Series Publication, 65SD531.

SAMPLE SOLUTION

```
*LIST

010 DIMENSION A(4),AI(4),RR(3),CR(3)
020 DATA A/0.,18.,-5.,1./,AI/0.,-6.,-3.,1./
030 CALL CD0WNH(A,AI,3,RR,CR)
040 D0 5 I=1,3
050 5 PRINT 10,I,RR(I),CR(I)
060 10 F0RMAT(5H0R00T,I2,10X,1P2E20.7)
070 D0 15 I=1,4
080 IM1=I-1
090 15 PRINT 20,IM1,A(I),AI(I)
100 20 F0RMAT(33H0RECONSTRUCTED C0EFFICIENT 0F X**,I2,1P2E18.7)
110 ST0P
120 END

READY

*RUN*;ZCOP2

R00T 1                    0.                    0.

R00T 2                    4.0000000E+00         1.9999999E+00

R00T 3                    0.                   -2.9999999E+00

RECONSTRUCTED C0EFFICIENT 0F X** 0      0.                    0.

RECONSTRUCTED C0EFFICIENT 0F X** 1      1.7999999E+01        -6.0000001E+00

RECONSTRUCTED C0EFFICIENT 0F X** 2     -5.0000000E+00        -3.0000000E+00

RECONSTRUCTED C0EFFICIENT 0F X** 3      1.0000000E+00         1.0000000E+00

PR0GRAM ST0P AT 110
*
```

ZEROES

This BASIC program locates values of X for any arbitrary function of X. Specifically, the program locates the values of X at which relative maximums and minimums of F (X) occur, and the values of X for which F (X) is zero, i. e., the zeros or roots of the function.

INSTRUCTIONS

Enter the data in the following format:

```
100 DATA XMIN,XMAX,ACC,INCR
200 LET Y = A function of X
RUN
```

where,

- XMIN, XMAX define the interval in which values of X are to be sought.
- ACC is the accuracy (in number of significant figures) to which the ZEROES of X and the maximum and minimum values of F (X) are to be estimated.
- INCR is the number of increments into which the total interval is to be divided for search purposes (try 50 to start).
- The function of X is any legitimate BASIC language expression involving the variable X.

SAMPLE PROBLEM

To illustrate how this program is used consider the sine function; it crosses the axis at 0 degrees and every 180 degrees thereafter, has maximums at 90 degrees and every 180 degrees thereafter, and has minimums at 270 degrees and every 180 degrees thereafter.

Since BASIC assumes all trigonometric functions are in radians, convert degrees to radians by multiplying X by π /180.

SAMPLE SOLUTION

```
*100 DATA 0,725,3,200
*200 LET Y=SIN(X*3.14159265/180)
*RUN
POINT-TYPE              F(X)              X

 ZERO                      0              0
  MAX                      1        90.00195

 ZERO                      0        180.0039
  MIN                     -1        270.0059

 ZERO                      0        360.0078
  MAX                      1        450.0098

 ZERO                      0        540.0117
  MIN                     -1        630.0137

 ZERO                      0        719.9873

READY
*
```

ZORP

This FORTRAN program approximates the roots of a polynomial of the form

$$P(z) = \sum_{n=0}^{I} A_n Z^n$$

where A_n is real.

INSTRUCTIONS

Instructions for the format of the input data are generated by the program, as shown in the Sample Problem. The program permits input errors to be corrected and generates the instructions required to make the corrections.

After the zeros of the polynomial have been approximated, the program reconstructs the polynomial coefficients using these approximations. Both the zeros and the reconstructed polynomial coefficients are printed by the program.

After the solution for the first polynomial has been computed, the program permits the user to define a new polynomial or discontinue the execution process.

RESTRICTIONS

The degree of the polynomial must not exceed 100.

The imaginary part of a root is set to zero if its magnitude is less than or equal to .0001 times the magnitude of the real part.

METHOD

The method used to approximate the zeros of the polynomial is a modified Downhill-Newton method*.

SAMPLE PROBLEM

Find the roots of the following polynomials:

$$X^4 + 18X^3 + 89X^2 + 72X - 180 = 0$$

$$X^{10} - 4X^9 - 3X^8 + 16X^7 + 15X^6 - 28X^5 - 21X^4 + 64X^3 + 44X^2 - 48X - 36 = 0$$

* Lilley, F. E., Zeroes Of A Polynomial, General Electric Company, Technical Information Series Publication, 65SD531.

SAMPLE SOLUTION

```
RUN
DØ YØU DESIRE USER INSTRUCTIØNS, TYPE YES ØR NØ
= YES
THIS PRØGRAM FINDS THE ZERØS ØF A PØLYNØMIAL ØF THE FØRM:

A(1)+A(2)*(Z)+A(3)*(Z**2)+...+A(N+1)*(Z**N)

WHERE THE DEGREE N CANNØT EXCEED 100.
CØEFFICIENTS ARE TYPED IN LØW ØRDER FIRST.
INPUT FØRMAT IS FREE FIELD: THE A(I) ARE REAL AND N IS INTEGER.

NØW YØU TRY IT

DEGREE
= 4
CØEFFICIENTS
= -180.,72.,89.,18.,1.
ANY CØRRECTIØNS, TYPE YES ØR NØ
= NØ
RØØT NØ.          REAL PART            CØMPLEX PART
   1          -0.30000002E+01          0.
   2           0.10000000E+01          0.
   3          -0.59999999E+01          0.
   4          -0.99999999E+01          0.

SUBSCRIPT     RECØNSTRUCTED CØEFFICIENTS
     1             -0.18000001E+03
     2              0.71999999E+02
     3              0.88999999E+02
     4              0.18000000E+02
     5              0.10000000E+01

DØ YØU WISH TØ SØLVE ANØTHER PØLYNØMIAL, TYPE YES ØR NØ
= YES
DEGREE
= 10
CØEFFICIENTS
= -36.,-48.,44.,64.,-21.,-28.,15.,16.,-3.,-4.,1.
ANY CØRRECTIØNS, TYPE YES ØR NØ
= NØ
RØØT NØ.          REAL PART            CØMPLEX PART
   1          -0.10021222E+01          0.
   2          -0.99893889E+00         -0.18491970E-02
   3          -0.99893889E+00          0.18491970E-02
   4           0.10000000E+01          0.
   5           0.99999997E+00         -0.99999999E+00
   6           0.99999997E+00          0.99999999E+00
   7          -0.10000000E+01          0.10000001E+01
   8          -0.10000000E+01         -0.10000001E+01
   9           0.30000001E+01          0.34526698E-03
  10           0.30000001E+01         -0.34526698E-03
```

```
SUBSCRIPT     RECØNSTRUCTED CØEFFICIENTS
    1              -0.36000009E+02
    2              -0.46000004E+02
    3               0.44000006E+02
    4               0.63999999E+02
    5              -0.21000008E+02
    6              -0.27999997E+02
    7               0.15000003E+02
    8               0.15999999E+02
    9              -0.29999999E+01
   10              -0.40000001E+01
   11               0.10000000E+01

DØ YØU WISH TØ SØLVE ANØTHER PØLYNØMIAL, TYPE YES ØR NØ
= NØ

PRØGRAM STØP AT 0
*
```

ZORP2

This FORTRAN subprogram finds the roots of a polynomial from its real coefficients.

INSTRUCTIONS

The calling sequence for this subprogram is:

CALL DOWNH(A, N, RR, CR)

where,

- A is the array of the real polynomial coefficients.
- N is the degree of the polynomial.
- RR is the array of the real parts of the roots.
- CR is the array of the imaginary parts of the roots.

The polynomial coefficients must be stored in ascending powers of the variable, i.e., constant term first.

RESTRICTIONS

The degree, N, must not exceed 25.

The imaginary part of a root is set to zero if its magnitude is less than or equal to .0001 times the magnitude of the real part.

METHOD

Modified Downhill-Newton scheme.

The coefficients are reconstructed from the computed roots and stored in A, thus destroying the original polynomial coefficients[1].

SAMPLE PROBLEM

Find the roots of the following polynomial:

$$X^5 + X^4 + X^3 + X^2 + X^1 + 1 = 0$$

[1] Lilley, F. E., Zeroes of a Polynomial, General Electric Company, Technical Information Series Publication, 65SD531.

SAMPLE SOLUTION

```
*LIST

010 DIMENSION A(6),RR(5),CR(5)
020 DATA A/1.,1.,1.,1.,1.,1./
030 CALL DOWNH(A,5,RR,CR)
040 DO 5 I=1,5
050 5 PRINT 10,I,RR(I),CR(I)
060 10 FORMAT(5H ROOT,I2,10X,1P2E20.7)
070 PRINT 20
080 20 FORMAT("0THE RECONSTRUCTED COEFFICIENTS OF THE POLYNOMIAL ARE:")
090 DO 25 I=1,6
100 25 PRINT 30, A(I)
110 30 FORMAT(20X,1PE20.7)
120 STOP
130 END

READY

*RUN *;ZORP2
ROOT 1                -9.9999998E-01         0.
ROOT 2                -4.9999998E-01          8.6602537E-01
ROOT 3                -4.9999998E-01         -8.6602537E-01
ROOT 4                 4.9999997E-01          8.6602540E-01
ROOT 5                 4.9999997E-01         -8.6602540E-01

THE RECONSTRUCTED COEFFICIENTS OF THE POLYNOMIAL ARE:
                          9.9999985E-01
                          9.9999994E-01
                          1.0000000E+00
                          9.9999999E-01
                          1.0000000E+00
                          1.0000000E+00

PROGRAM STOP AT 120
*
```

Honeywell Bull

HONEYWELL INFORMATION SYSTEMS

DA43 - Rev. O

Ref. : 19.53.106 A