AECL-6076 (Rev. D)

ATOMIC ENERGY
OF CANADA LIMITED

L'ÉNERGIE ATOMIQUE
DU CANADA LIMITÉE

AELIB USERS' MANUAL

Manuel des utilisateurs de AELIB

L.E. EVANS and E.A. OKAZAKI

Chalk River Nuclear Laboratories   Laboratoires nucléaires de Chalk River

Chalk River, Ontario, Canada K0J 1J0

September 1978 septembre

Last Revision 1983 June   Revisé 1983 juin

REVISION RECORD

DATE	REVISION	DESCRIPTION
September 1978	Original	Original printing
July 1979	A	New writeups 2-3-11, 2-5-43, 2-5-73, 2-5-74 in Part B. Modifications to existing material in Part A and Part B.
April 1980	B	New Part A. New writeups 2-5-21, 2-5-32, 2-5-33, 2-5-36, 2-5-60, 2-5-61, 2-5-76, 2-5-77 in Part B. Modifications to existing material in Part A and Part B.
May 1981	C	Modifications to existing material in Part A and Part B.
June 1983	D	New Part A. New Section 1-4. New writeups 1-5-01, 1-5-11, 1-11-16, 1-11-24, 1-11-25, 1-11-30, 2-5-44. Modifications to existing material in Part B. 1-10-20 removed.

ATOMIC ENERGY OF CANADA LIMITED

AELIB USERS' MANUAL

Compiled and Edited by

L.E. Evans
and
E.A. Okazaki

With Contributions from

J.M. Blair
M.B. Carver
G.H. Keech
G.L. Klawitter
B.E. Purcell
W.N. Selander
C.A. Wills
P.Y. Wong

Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0
September 1978
Revised July 1979
Revised April 1980
Revised May 1981
Revised June 1983

AECL-6076

ATOMIC ENERGY OF CANADA LIMITED

AELIB USERS' MANUAL

Compiled and Edited by

L.E. Evans
and
E.A. Okazaki

With Contributions from

J.M. Blair
M.B. Carver
G.H. Keech
G.L. Klawitter
B.E. Purcell
W.N. Selander
C.A. Wills
P.Y. Wong

ABSTRACT

This report is an updatable manual for users of AELIB, the AECL Library of FORTRAN-callable routines for the CDC computers in use at AECL. It provides general advice on the use of this library and detailed information on the selection and usage of particular library routines. This report replaces the unpublished internal report CRNL-486.

Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0
September 1978
Revised July 1979
Revised April 1980
Revised May 1981
Revised June 1983

AECL-6076

L'ENERGIE ATOMIQUE DU CANADA, LIMITEE

MANUEL DES UTILISATEURS D'AELIB

Compilé et édité par

L.E. Evans et E.A. Okazaki

avec la collaboration de

J.M. Blair
M.B. Carver
G.H. Keech
G.L. Klawitter
B.E. Purcell
W.N. Selander
C.A. Wills
P.Y. Wong

Résumé

Le présent rapport est un manuel pouvant être mis à jour. Il est destiné aux utilisateurs d'AELIB, bibliothèque des sous-programmes FORTRAN de l'EACL employés dans les ordinateurs CDC en service à l'EACL. Il donne des conseils généraux en ce qui concerne l'usage de cette bibliothèque et des informations détaillées concernant la sèlection et l'usage de certains sous-programmes de bibliothèque. Le présent rapport remplace le rapport interne CRNL-486 non publié.

Laboratoires nucléaires de Chalk River
Chalk River, Ontario, Canada KOJ 1JO

Septembre 1978
Révisé en juillet 1979
Révisé en avril 1980
Révisé en mai 1981
Révisé en juin 1983

AECL-6076

TABLE OF CONTENTS

PART A - The Use of AELIB

1. AELIB OVERVIEW

(a) General Information

AELIB is a collection (i.e., a library) of subroutines and functions which may be used from any FORTRAN program run on the CDC 6600/175/720 system in use at AECL. These subroutines are maintained and updated when necessary by the Mathematics and Computation Branch applications programming staff. In support of this library, current algorithms as published in the literature are reviewed continuously and the library is modified to reflect the state of relevant topics and to meet the needs of the library users. Library changes are therefore inevitable. We attempt, however, to make such changes at most three times per year. Users are informed of any library changes well in advance via the Newsletter[1] and, for continuity, source language card decks are provided without programming support for any routines to be removed. Questions and suggestions about AELIB policy and operation should be directed to the site consultant, or to the Newsletter. Specific comments on this manual may be made by filling out and returning the Comment Sheet inside the back cover.

This report is organized in two parts. Part A contains information on the use of AELIB that will be of interest to all users, with more specialized information provided in appendices. Part B contains detailed documentation for each of the AELIB routines and for a small number of other FORTRAN-callable routines which are local additions to other system libraries. Normally the user is not concerned about identifying the system library from which routines are loaded, but there are cases in which this information is necessary. The FTN4 version of the AELIB library is resident on the system library AELIB, while the FTN5 version is on the system library AELIB5. Writeups for routines that are not part of the AELIB collection include the name of the library on which they reside.

This report and the FORTRAN reference manuals[2,3] together describe all CDC and locally supported FORTRAN-callable subroutines. Appendix 1 provides a keyword index to all the routines written up in this report. Appendix 2 describes the other library collections that are available, including IMSL of International Mathematical and Statistical Libraries, Inc.

AELIB routines are available to any user of the Computing Centre. However, the documentation contained in this report does assume a certain level of user sophistication. Users having difficulty using AELIB should follow the instructions in the flow chart of Appendix 3.

(b) Support of AELIB in Dual-Compiler/ Dual-Operating System Environment

Because of fundamental system differences between NOS and NOS/BE and between FTN4 and FTN5, four separate binary versions of the AELIB collection are supported, one for each combination of compiler (FTN4 or FTN5) and operating system (NOS or NOS/BE). AELIB is the name of the FTN4 version of the library on NOS and NOS/BE and AELIB5 is the name of the FTN5 version on both systems. On NOS/BE, and on NOS on the CRNL CYBER 720, the appropriate version of AELIB is automatically invoked by the compiler, whereas on NOS on the Sheridan Park CYBER 720, this is not the case. Specific instructions for accessing AELIB routines on NOS are given in Section 4.

In almost all cases the library routines function in the same way in the four versions and, except for minor compiler or system differences, should produce the same computed results. Appendix 6 provides general information on the type of compiler and system differences to be expected. For those routines for which not all four versions are supported or for which there are major compiler or system differences among the various versions, a Support Differences section is included in the write-ups in Part B of this manual.

As of 1983 February full library support is provided for the FTN5 versions of AELIB, with parallel support being provided for the FTN4 versions where practicable. The source code for AELIB is supported as a single UPDATE program library using the UPDATE *IF directive to select the portions of the code that differ for the four versions. Section 5 provides specific instructions for the retrieval of source code.

2. HOW TO FIND A SUBROUTINE IN AELIB

Each subroutine in AELIB has a write-up describing its use, and these write-ups are organized in Part B of this report according to the following subject matter classification scheme:

SECTION 1 - MATHEMATICAL ROUTINES

1- 1	MATHEMATICAL AND PHYSICAL CONSTANTS
1- 2	SPECIAL MATHEMATICAL AND STATISTICAL FUNCTIONS
1- 3	PHYSICS FUNCTIONS
1- 4	ENGINEERING FUNCTIONS
1- 5	RANDOM NUMBER GENERATION
1- 6	STATISTICAL TESTING (FOR FUTURE USE)
1- 7	MATRIX AND VECTOR MANIPULATION
1- 8	LINEAR EQUATIONS AND MATRIX INVERSION
1- 9	MATRIX EIGENSYSTEM ANALYSIS
1-10	SOLUTION OF NONLINEAR EQUATIONS AND FUNCTION EXTREMA
1-11	FITTING USER SPECIFIED FUNCTIONS TO DATA

1-12	DATA SMOOTHING
1-13	TIME SERIES ANALYSIS
1-14	INTERPOLATION
1-15	DIFFERENTIATION
1-16	INTEGRATION OF USER SUPPLIED FUNCTIONS
1-17	INTEGRATION OF ORDINARY DIFFERENTIAL EQUATIONS
1-18	INTEGRATION OF PARTIAL DIFF EQNS (FOR FUTURE USE)

SECTION 2 - NON-MATHEMATICAL AND UTILITY ROUTINES

2- 1	PLOTTING
2- 2	INPUT/OUTPUT ROUTINES (EXCLUDING PLOTTING)
2- 3	TABULAR DATA MANIPULATION
2- 4	CHARACTER AND BIT STRING MANIPULATIONS
2- 5	SYSTEM FACILITIES

Write-ups are numbered X-Y-Z where X-Y is the number of the subject matter classification and Z is the number of the routine in that subject. Each subject matter group of write-ups is preceded by an index and, in most cases, an overview of the section with usage recommendations and references to other libraries.

One way, then, to find a routine in AELIB or to find whether there are any routines covering a certain subject is to browse through the appropriate section of write-ups. As an alternative, the keyword index in Appendix 1 may be used. This index provides the name, number, title and characteristic keywords, of routines in alphabetical order by keyword. Since routine names are keywords, this index may also be used to quickly locate particular routines.

3. DOCUMENTATION FORMAT AND CONVENTIONS

The standard format for write-ups is as follows, with optional specifications in parentheses:

Title
(Introduction)
Calling Sequence or Entry
(Additional Entry Information)
Exits, both Normal and Error
Routines Called, if any
Common Blocks Used, if any

System Residence, if not AELIB or AELIB5
Approximate Storage Requirements, if in excess of 1000_8 words
Accuracy, for Mathematical Routines
Speed, for Mathematical Routines
Program Example
Support Differences, if any
External Version, if supported
References, if any
Author and Revision History

A few write-ups do not yet meet this standard. These will be either removed or upgraded in future updates to this manual. Method descriptions have not been included in the documentation. Contact the AELIB librarian via your site consultant, or look at a source listing (see Section 5) for details on the methods used. The following conventions are used for write-ups:

(1) Output variables in the calling sequence are identified by underlining, either as ___ or ~. Any variable returning a value to the calling routine is an output variable. Those variables used only for output are underlined with ___ while those that are used as input and output are underlined with ~.

(2) Subroutine entries are specified in the form CALL SUB(.....). Function entries are specified as R=FUNC(...),I=FUNC(...) or C=FUNC(...) depending on whether the function is real, integer or complex valued. Note that functions may also be used directly from FORTRAN expressions, for example, X=X*FUNC(...)+Y.

(3) Upper case letters are used for the names of calling sequence variables while corresponding lower case letters may be used in the write-up to describe the mathematical quantity represented by that variable.

(4) Commonplace FORTRAN system routines such as SQRT and SIN and low level system routines will not be included in the "ROUTINES CALLED" section.

(5) Included in the speed section are the machine (CYBER 170 Model 175, CDC 6600, or CYBER 170 Model 720) and date on which the timing tests were run. Use of FTN4 is assumed for tests done prior to 1983 February. Tests done after that time will identify the compiler as well as the computer used. For NOS/BE job card time estimates, 175 timing figures should be used. (Jobs normally run about 2.5 times faster on the 175 than on the 6600, so to get 175 timing estimates from 6600 timing estimates, divide by 2.5). If you require more detailed information about the relevance of old timing data, contact the AELIB librarian via your site consultant.

4. HOW TO USE AN AELIB SUBROUTINE

All routines described or referenced in this report are resident on system libraries and are provided automatically at load time to NOS/BE programs which call them. To use any of these routines on NOS/BE, then, just compile and execute a FTN4 or FTN5 program which calls them according to the documentation specifications. The same applies to the NOS system on the CRNL CYBER 720.

On the NOS system on the Sheridan Park CYBER 720, however, the FORTRAN system routines are loaded automatically but explicit control statements are required to access the AELIB collection of subroutines.

On the Engineering Company CYBER 170 Model 720 at Sheridan Park, use:

```
ATTACH,AELIB/UN=PUBCRNL.           with the FTN4 compiler
LIBRARY,AELIB.

or

ATTACH,AELIB5/UN=PUBCRNL.          with the FTN5 compiler
LIBRARY,AELIB5.
```

5. HOW TO GET SOURCE CODE FOR LIBRARY ROUTINES

Source code for library routines is maintained on UPDATE program libraries which parallel the system object code libraries from which the routines are loaded. Source code for most routines on libraries other than AELIB cannot be made available because of software licensing agreements with Control Data Corporation and IMSL. Requests for source code for these routines should be directed to the site consultant.

Source code for all four versions of AELIB and a limited number of external versions of system dependent routines suitable for export to other computing centres are maintained on a single UPDATE program library and a current copy of this library is available on the NOS/BE system on a nine-track labelled magnetic tape with visual serial number (VSN) AELIBS. NOS/BE compiled source code for a particular version of an AELIB routine except for PLOT† may be obtained by specifying the appropriate UPDATE *DEFINE directives from Table 1 in one of the NOS/BE control statement sequences as follows:

† Source code for PLOT should not be required. PLOT is part of our local system software and is written in SYMPL, a systems programming language.

(1) To obtain a listing, use

```
JOB,Bnnn-nnnnn,T10,IO20,NT1.
VSN(OLDPL=AELIBS)
LABEL(OLDPL,D=PE,R,L=AELIBPL,NORING)
UPDATE(Q)
FTN(I,R=3) or FTN5(I,LO=R/A/M/S,OPT=2)
7/8/9
*DEFINE directive
*COMPILE deckname
```

(2) To get a source card deck in addition to the listing replace the UPDATE and FTN cards in (1) by

```
UPDATE(Q,C=PUNCH)
FTN(I=PUNCH,R=3)   or   FTN5(I=PUNCH,LO=R/A/M/S,OPT=2)
```

Deckname is the name of the routine or its main entry point. This name is provided on the special page footing used in Part B of this manual.

TABLE 1 - UPDATE *DEFINE Directive Required for AELIB Source Code

	FTN4	FTN5
Internal NOS/BE	none	AELIB5
Internal NOS	NOS	NOS,AELIB5
External	EXTERNAL	AELIB5,EXTERNAL

Some library "routines" are modular in form and, therefore, actually consist of several subroutines. In these cases, source code for all modules is provided under the deckname of the main routine. If some modules are shared by a group of similar AELIB routines, the source code for the common routines resides with one main routine and is usually clearly referenced by comments in the others. To help users to interpret loader maps, a cross-reference list of these extra subroutine modules is provided in Appendix 4.

While free access is allowed to the source code for AELIB routines, the user requesting source code must be aware of the following:

(1) Support is provided only for routines resident on the system libraries. Users who use private versions of library routines do so at their own risk.

(2) Frequently applications programs that use AELIB subroutines are sent to other computing centres. The export of AELIB source code to non-AECL users is permitted in these cases subject to the following restrictions:

(a) AELIB code must not be transferred to third parties;

(b) AELIB routines must not be used to generate commercial revenue for outside organizations;

(c) Publications in which AELIB subroutines are mentioned must acknowledge their source.

(3) Several AELIB routines, particularly in the non-mathematical areas, use CRNL system dependent features. To facilitate the export of user programs and the necessary AELIB code to other computing centres, it is AELIB policy to provide, where possible, an external version of system dependent routines. This feature is partially supported now by providing for some routines, CDC compatible non-CRNL system dependent code under the control of the UPDATE *IF directive. To access the external code, use the *DEFINE EXTERNAL directive in the job that retrieves the source code. A brief explanation of the external version is provided in the write-ups for those routines for which this feature is supported.

In particular the *DEFINE EXTERNAL directive will remove calls to the FTN4 usage statistics routine LIBSTAT (2-5-75) and will retrieve a version of the system dependent error message processing routine AELERR (2-5-71) that prints only error message numbers.

6. HOW TO KEEP UP WITH AELIB DEVELOPMENT

As explained in Section 1, AELIB is continually under review and library changes are to be expected. Such changes are always announced in the Newsletter along with frequent special interest articles on some aspect of library development. Also, the system bulletin AELIB is provided to supplement this library manual by providing the following information:

- the current edition of the library manual
- current contents of the system libraries
- a summary of current developments
- a list of recent library changes

A copy of this bulletin, which may be printed by executing the control card

```
SYSBULL(AELIB)
```

should be filed in the front of this manual for easy reference.

7. HOW TO MAKE SURE THAT YOUR MANUAL IS UP TO DATE

We will provide manual updates for all library changes. It is your responsibility to make sure that the manual you are using is up to date. CANDU Operations (Sheridan Park) and WNRE users have their own update distribution service. For other users, the update distribution works as follows:

> When you first obtain your manual, simply remove, fill out, and send us the form at the beginning of the manual, and we will put your name on our update distribution list. When update packages are prepared, they will be automatically sent to you for insertion in your manual.

L.E. Evans
1983 June

APPENDIX 1

THE KEYWORD INDEX

The keyword index that follows can be used to quickly locate the documentation for FORTRAN-callable subroutines in this manual. The index lists these subroutines in alphabetic order of their keywords in the following format:

KEYWORD	TITLE	ROUTINE
⋮	⋮	⋮

Notes:

(1) KEYWORD

For subroutines described in this manual, the keywords consist of all entry point names plus relevant subject matter keywords (such as Bessel Function, Integration).

(2) ROUTINE

Subroutines are identified in this field by the name of their main or generic entry and the location of their documentation. Numbers refer to specific write ups in Part B of this report.

KEYWORD	TITLE	ROUTINE	
ACCOUNTING PARAMETERS	USER ACCOUNTING PARAMETERS	USERAP	2-5-44
ADD	SERIES SUMMATION FUNCTIONS	TSUM	1-7-00
ADD	ADD SUBTRACT MULTIPLY COMPLEX MATRICES	CMMULT	1-7-02
ADD	ADD SUBTRACT MULTIPLY REAL MATRICES	MMULT	1-7-01
AEFREQ	ERROR MESSAGE PRINTING CONTROL - OLD	AEFREQ	2-5-72
AEISER	ISSUE ERROR MESSAGE FROM COMPASS	AEISER	2-5-76
AELERR	PRINT ERROR MESSAGE WITH TRACEBACK - OLD	AELERR	2-5-71
AELERTX	ERROR MESSAGE TEXT - OLD	AELERTX	2-5-70
AGAUSS	ADAPTIVE GAUSS INTEGRATION	AGAUSS	1-16-02
AIKINT	AITKEN INTERPOLATION	AIKINT	1-14-02
AINERF	INVERSE ERROR FUNCTION	AINERF	1-2-06
AINERFC	INVERSE ERROR FUNCTION	AINERF	1-2-06
AIRY FUNCTION	AIRY FUNCTION AND DERIVATIVE, REAL ARG	JAIRY	1-2-43
AIOBES	MODIFIED BESSEL FUNCTION I0, REAL ARG	AIOBES	1-2-34
AI1BES	MODIFIED BESSEL FUNCTION I1, REAL ARG	AI1BES	1-2-35
AJYBES	BESSEL FUNCTIONS J0,J1,Y0,Y1, REAL ARG	AJYBES	1-2-30
AKI1	BICKLEY FUNCTION KI1	AKI1	1-2-50
AKI3	BICKLEY FUNCTION KI3	AKI3	1-2-51
AKOBES	MODIFIED BESSEL FUNCTION K0, REAL ARG	AKOBES	1-2-36
AK1BES	MODIFIED BESSEL FUNCTION K1, REAL ARG	AK1BES	1-2-37
ALAGIN	CUBIC LAGRANGE INTERPOLATION	ALAGIN	1-14-01
ALEGEND	ASSOCIATED LEGENDRE FUNCTION	ALEGEND	1-2-61
ALERDM	DUMP ERROR MESSAGE STATUS TABLE	ALERDM	2-5-73
ALEROF	DISABLE SELECTED AELIB ERROR MESSAGES	ALEROF	2-5-73
ALERON	RE-ENABLE SELECTED AELIB ERROR MESSAGES	ALERON	2-5-73
ALERPR	RETURN PRINT STATUS FOR ERROR MESSAGES	ALERPR	2-5-73
ARSINH	INVERSE HYPERBOLIC SINE	ARSINH	1-2-90
ASUM	SERIES SUMMATION FUNCTIONS	TSUM	1-7-00
AXIS	CALCOMP AXIS PLOTTING ROUTINE	AXIS	2-1-67
AXLGX	SEMI-LOG X AXIS LABELLING ROUTINE	AXLGX	2-1-10
AXLGXM	SEMI-LOG X AXIS LABELLING ROUTINE	AXLGX	2-1-10
AXLGXY	LOG-LOG X AND Y AXES LABELLING ROUTINE	AXLGXY	2-1-10
AXLGXYM	LOG-LOG X AND Y AXES LABELLING ROUTINE	AXLGXY	2-1-10
AXLGY	SEMI-LOG Y AXIS LABELLING ROUTINE	AXLGY	2-1-10
AXLGYM	SEMI-LOG Y AXIS LABELLING ROUTINE	AXLGY	2-1-10
AXLIN	LINEAR AXES LABELLING ROUTINE	AXLIN	2-1-10
AXLINM	LINEAR AXES LABELLING ROUTINE	AXLIN	2-1-10
BESGEN	BESSEL FUNCTIONS, INTGR ORDER, REAL ARG	BESGEN	1-2-38
BESSEL FUNCTION	BESSEL FUNCTIONS, INTGR ORDER, REAL ARG	BESGEN	1-2-38
BESSEL FUNCTION	BESSEL FUNCTIONS J0,J1,Y0,Y1, REAL ARG	AJYBES	1-2-30
BESSEL FUNCTION	BESSEL FUNCTION JP(Z),REAL P, COMPLEX Z	JBESCPX	1-2-42
BESSEL FUNCTION	MODIFIED BESSEL FUNCTION I0, REAL ARG	AIOBES	1-2-34
BESSEL FUNCTION	I BESSEL FUNCTION OF REAL ORDER AND ARG	IBESS	1-2-40
BESSEL FUNCTION	MODIFIED BESSEL FUNCTION K0, REAL ARG	AKOBES	1-2-36
BESSEL FUNCTION	BESSEL FUNCTION Y0 FOR COMPLEX ARG	CBESY0	1-2-44
BESSEL FUNCTION	MODIFIED BESSEL FUNCTION I1, REAL ARG	AI1BES	1-2-35
BESSEL FUNCTION	J BESSEL FUNCTION OF REAL ORDER AND ARG	JBESS	1-2-39
BESSEL FUNCTION	MODIFIED BESSEL FUNCTION K1, REAL ARG	AK1BES	1-2-37
BESSEL FUNCTION	BESSEL FUNCTION Y1 FOR COMPLEX ARG	CBESY1	1-2-45
BESSEL FUNCTION	BESSEL FUNCTIONS IN,JN FOR COMPLEX ARG	COMBES	1-2-41

KEYWORD	TITLE	ROUTINE	
BICKLEY FUNCTION	BICKLEY FUNCTION KIN	KIN	1-2-52
BICKLEY FUNCTION	BICKLEY FUNCTION KI3	AKI3	1-2-51
BICKLEY FUNCTION	BICKLEY FUNCTION KI1	AKI1	1-2-50
BIT MANIPULATION	SHIFTING	SHIFTB	2-4-20
BIT MANIPULATION	PACK OR UNPACK BIT STRINGS	PAKUNP	2-4-10
BIT MANIPULATION	BYTE EXTRACTION	LBYT	2-4-00
BIT MANIPULATION	BYTE REPLACEMENT	SBYT	2-4-01
BIT MANIPULATION	BIT EXTRACTION AND REPLACEMENT	BITS	2-4-02
BITS	BIT EXTRACTION AND REPLACEMENT	BITS	2-4-02
CADRE	EXTRAPOLATIVE ROMBERG INTEGRATION	CADRE	1-16-00
CBESY0	BESSEL FUNCTION Y0 FOR COMPLEX ARG	CBESY0	1-2-44
CBESY1	BESSEL FUNCTION Y1 FOR COMPLEX ARG	CBESY1	1-2-45
CCL REGISTER ACCESS	FETCH CCL REGISTER VALUE	IGETCCL	2-5-60
CCL REGISTER ACCESS	SET VALUE IN CCL REGISTER	SETCCL	2-5-61
CDC	CDC MACHINE DEPENDENT CONSTANTS	FLPT	1-1-10
CDET	DETERMINANT OF A COMPLEX MATRIX	CDET	1-7-04
CEL3	COMPLETE ELLIPTIC INTEGRAL OF 3RD KIND	CEL3	1-2-22
CENTRE OF CURVATURE	APPROX RADIUS AND CENTRE OF CURVATURE	RADCURV	1-11-30
CGAMLN	GAMMA AND LN(GAMMA) FOR COMPLEX ARG	CGAMMA	1-2-12
CGAMMA	GAMMA AND LN(GAMMA) FOR COMPLEX ARG	CGAMMA	1-2-12
CHAR	BIT EXTRACTION AND REPLACEMENT	BITS	2-4-02
CHI SQUARE PROBLTY	INCOMPLETE GAMMA FUNCTION	GAMIN	1-2-13
CHISQ	CHI-SQUARE PROBABILITY FUNCTION	CHISQ	1-2-80
CHNGEC	DYNAMIC ALLOCATION OF ECS	CHNGEC	2-5-36
CHNGEX	MEMORY MANAGEMENT (EXPANDABLE BLOCKS)	CHNGEX	2-5-33
CHNGFX	MEMORY MANAGEMENT (FIXED-LENGTH BLOCKS)	CHNGFX	2-5-32
CHNGSCM	FIELD LENGTH MANAGEMENT -STATIC MODE	CHNGSCM	2-5-30
CIPLOT	PLOTTING CONFIDENCE INTERVALS	CIPLOT	1-11-16
CLEBSCH	CLEBSCH-GORDAN COEFFICIENT	CLEBSCH	1-3-20
CMADD	ADD SUBTRACT MULTIPLY COMPLEX MATRICES	CMMULT	1-7-02
CMMULT	ADD SUBTRACT MULTIPLY COMPLEX MATRICES	CMMULT	1-7-02
CMSUB	ADD SUBTRACT MULTIPLY COMPLEX MATRICES	CMMULT	1-7-02
COMBES	BESSEL FUNCTIONS IN,JN FOR COMPLEX ARG	COMBES	1-2-41
COMKTM	DEFINE A POLYNOMIAL OF DEG I IN (X1,..XK	COMKTM	1-11-12
CONFIDENCE INTERVALS	PLOTTING CONFIDENCE INTERVALS	CIPLOT	1-11-16
CONTOUM	CONTOUR SURFACE PLOTTING	CONTOUR	2-1-35
CONTOUR	CONTOUR SURFACE PLOTTING	CONTOUR	2-1-35
CONVOLUTION	GAUSSIAN/LORENTZIAN CONVOLUTION	GLINT	1-3-01
COSIMP	ADAPTIVE SIMPSONS INTEGRATION	COSIMP	1-16-01
COULOMB	COULOMB WAVE FUNCTIONS	COULOMB	1-3-10
CSOLVEQ	SOLVE A COMPLEX LINEAR SYSTEM	CSOLVEQ	1-8-01
CUBIC	ROOTS OF A CUBIC EQUATION	CUBIC	1-10-00
DATA STORAGE	DATA MANAGEMENT PACKAGE	DATPAK	2-2-10
DATPAK	DATA MANAGEMENT PACKAGE	DATPAK	2-2-10
DESORT	SORTING DATA WITH TAG	SORTAG	2-3-10
DET	DETERMINANT OF A REAL MATRIX	DET	1-7-03
DETERMINANT	DETERMINANT OF A COMPLEX MATRIX	CDET	1-7-04
DETERMINANT	DETERMINANT OF A REAL MATRIX	DET	1-7-03
DIFFERENTIAL EQNS	SHORT RUNGE-KUTTA INTEGRATION	RKONE	1-17-00
DIFFERENTIAL EQNS	INTEGRATION USING GEARS ALGORITHM	STIFFZ	1-17-20

KEYWORD	TITLE	ROUTINE	
DIFFERENTIAL EQNS	INTEGRATION USING RUNGE-KUTTA-FEHLBERG	RKFINT	1-17-10
DIFFERENTIATION	DIFFERENTIATION OF DATA BY SPLINES	HDERIV	1-12-10
DIFFERENTIATION	FIRST DERIVATIVES OF A TABULATED FUNC	DUDX	1-15-00
DIFFERENTIATION	SECOND DERIVATIVES OF A TABULATED FUNC	DUDXX	1-15-00
DISTRIBUTION	CHI-SQUARE PROBABILITY FUNCTION	CHISQ	1-2-80
DISTRIBUTION	STUDENTS T-DISTRIBUTION FUNCTION	TDISTN	1-2-81
DMPFET	DUMP FIT,FET AND BUFFERS	DMPFET	2-5-00
DUBLINT	GAUSSIAN DOUBLE INTEGRATION	DUBLINT	1-16-20
DUDX	FIRST DERIVATIVES OF A TABULATED FUNC	DUDX	1-15-00
DUDXX	SECOND DERIVATIVES OF A TABULATED FUNC	DUDXX	1-15-00
DUDXX1	SECOND DERIVATIVES OF A TABULATED FUNC	DUDXX	1-15-00
DUDX1	FIRST DERIVATIVES OF A TABULATED FUNC	DUDX	1-15-00
DUMP	DUMP FIT,FET AND BUFFERS	DMPFET	2-5-00
DUMPOFF	SIMPLE PLOTTING ROUTINE	SIMPLT	2-1-20
DUMPON	SIMPLE PLOTTING ROUTINE	SIMPLT	2-1-20
DYNAMIC VISCOSITY	DYNAMIC VISCOSITY OF LIGHT WATER	VISLW	1-4-04
ECS ADDRESSING	USE ECS ADDRESSES GREATER THAN 2**17	MOVLUV	2-5-35
ECS MANAGEMENT	DYNAMIC ALLOCATION OF ECS	CHNGEC	2-5-36
EI	EXPONENTIAL INTEGRAL	EI	1-2-00
EIGCGEN	EIGENSYSTEM OF COMPLEX GENERAL MATRIX	EIGCGEN	1-9-30
EIGCHER	EIGENSYSTEM OF COMPLEX HERMITIAN MATRIX	EIGCHER	1-9-20
EIGENANALYSIS	EIGENSYSTEM OF A REAL SYMMETRIC MATRIX	EIGRSYM	1-9-00
EIGENANALYSIS	EIGENSYSTEM OF COMPLEX GENERAL MATRIX	EIGCGEN	1-9-30
EIGENANALYSIS	EIGENSYSTEM OF A REAL GENERAL MATRIX	EIGRGEN	1-9-10
EIGENANALYSIS	EIGENSYSTEM OF COMPLEX HERMITIAN MATRIX	EIGCHER	1-9-20
EIGENVALUE	EIGENSYSTEM OF A REAL SYMMETRIC MATRIX	EIGRSYM	1-9-00
EIGENVALUE	EIGENSYSTEM OF COMPLEX GENERAL MATRIX	EIGCGEN	1-9-30
EIGENVALUE	EIGENSYSTEM OF A REAL GENERAL MATRIX	EIGRGEN	1-9-10
EIGENVALUE	EIGENSYSTEM OF COMPLEX HERMITIAN MATRIX	EIGCHER	1-9-20
EIGENVECTOR	EIGENSYSTEM OF A REAL SYMMETRIC MATRIX	EIGRSYM	1-9-00
EIGENVECTOR	EIGENSYSTEM OF COMPLEX GENERAL MATRIX	EIGCGEN	1-9-30
EIGENVECTOR	EIGENSYSTEM OF A REAL GENERAL MATRIX	EIGRGEN	1-9-10
EIGENVECTOR	EIGENSYSTEM OF COMPLEX HERMITIAN MATRIX	EIGCHER	1-9-20
EIGRGEN	EIGENSYSTEM OF A REAL GENERAL MATRIX	EIGRGEN	1-9-10
EIGRSYM	EIGENSYSTEM OF A REAL SYMMETRIC MATRIX	EIGRSYM	1-9-00
EINCO	INCOMPLETE ELLIPTIC INTEGRALS 1 AND 2	EINCO	1-2-23
ELE	COMPLETE ELLIPTIC INTEGRAL OF 2ND KIND	ELE	1-2-21
ELECOMP	COMPLETE ELLIPTIC INTEGRAL OF 2ND KIND	ELE	1-2-21
ELK	COMPLETE ELLIPTIC INTEGRAL OF 1ST KIND	ELK	1-2-20
ELKCOMP	COMPLETE ELLIPTIC INTEGRAL OF 1ST KIND	ELK	1-2-20
ELLIPTIC INTEGRAL	INCOMPLETE ELLIPTIC INTEGRALS 1 AND 2	EINCO	1-2-23
ELLIPTIC INTEGRAL	COMPLETE ELLIPTIC INTEGRAL OF 1ST KIND	ELK	1-2-20
ELLIPTIC INTEGRAL	COMPLETE ELLIPTIC INTEGRAL OF 3RD KIND	CEL3	1-2-22
ELLIPTIC INTEGRAL	COMPLETE ELLIPTIC INTEGRAL OF 2ND KIND	ELE	1-2-21
EL3COMP	COMPLETE ELLIPTIC INTEGRAL OF 3RD KIND	CEL3	1-2-22
EN	EXPONENTIAL INTEGRAL	EN	1-2-01
ERROR FUNCTION	INVERSE ERROR FUNCTION	AINERF	1-2-06
ERROR FUNCTION DIFF	RESONANCE INTEGRAL	RESIN	1-3-00
ERROR MESSAGES	DUMP ERROR MESSAGE STATUS TABLE	ALERDM	2-5-73
ERROR MESSAGES	RETURN PRINT STATUS FOR ERROR MESSAGES	ALERPR	2-5-73

KEYWORD	TITLE	ROUTINE	
INTEGRATION	INTEGRATION USING GEARS ALGORITHM	STIFFZ	1-17-20
INTEGRATION	EXTRAPOLATIVE ROMBERG INTEGRATION	CADRE	1-16-00
INTEGRATION	GAUSSIAN DOUBLE INTEGRATION	DUBLINT	1-16-20
INTEGRATION	ADAPTIVE SIMPSONS INTEGRATION	COSIMP	1-16-01
INTEGRATION	INTEGRATION OF TABULAR DATA	SPLINT	1-16-30
INTEGRATION	ADAPTIVE GAUSS INTEGRATION	AGAUSS	1-16-02
INTEGRATION	GAUSS-LAGUERRE INTEGRATION	GALA	1-16-11
INTEGRATION	SHORT RUNGE-KUTTA INTEGRATION	RKONE	1-17-00
INTEGRATION	GAUSS-JACOBI INTEGRATION	GAJAC	1-16-13
INTEGRATION	INTEGRATION USING RUNGE-KUTTA-FEHLBERG	RKFINT	1-17-10
INTEGRATION	FILON INTEGRATION	FILON	1-16-10
INTEGRATION	GAUSS-HERMITE INTEGRATION	GAHER	1-16-12
INTERPOLATION	INTERPOLATION OF EQUALLY SPACED DATA	SPINT	1-14-00
INTERPOLATION	AITKEN INTERPOLATION	AIKINT	1-14-02
INTERPOLATION	MULTIVARIATE INTERPOLATION	INPOL	1-14-10
INTERPOLATION	CUBIC LAGRANGE INTERPOLATION	ALAGIN	1-14-01
INVERSE	INVERSE ERROR FUNCTION	AINERF	1-2-06
ITSUM	SERIES SUMMATION FUNCTIONS	TSUM	1-7-00
JACOB	EVALUATE AND PACK A SPARSE JACOBIAN	JACOB	1-7-11
JAIRY	AIRY FUNCTION AND DERIVATIVE, REAL ARG	JAIRY	1-2-43
JBESCPX	BESSEL FUNCTION JP(Z),REAL P, COMPLEX Z	JBESCPX	1-2-42
JBESS	J BESSEL FUNCTION OF REAL ORDER AND ARG	JBESS	1-2-39
JOB PARAMETERS	CENTRAL PROCESSOR TIME USED OR LEFT	XTIME	2-5-40
JOB PARAMETERS	OBTAIN CURRENT JOBNAME	JOBNAME	2-5-43
JOB PARAMETERS	INPUT/OUTPUT TIME USED OR LEFT	TIMEIO	2-5-41
JOB PARAMETERS	MAINFRAME IDENTIFICATION ROUTINE	MFID	2-5-42
JOBNAME	OBTAIN CURRENT JOBNAME	JOBNAME	2-5-43
KIN	BICKLEY FUNCTION KIN	KIN	1-2-52
LBYT	BYTE EXTRACTION	LBYT	2-4-00
LEAST SQUARES	LEAST SQUARES FIT TO A POLYNOMIAL IN X	POLFIT	1-11-00
LEAST SQUARES	NL2SOL INTERFACE	NL2INT	1-11-24
LEAST SQUARES	FIT A POLYNOMIAL OF DEG I IN (X1,...XK)	PRFIT	1-11-10
LEAST SQUARES	FIT A GIVEN POLYNOMIAL IN (X1,...XK)	POLREG	1-11-11
LEAST SQUARES	MINIMIZE SUM OF SQUARES OF DIFF FUNC	NL2SOL	1-11-25
LEAST SQUARES	NONLINEAR LEAST SQUARES FIT TO F(X1,.XK)	MLSQQ	1-11-20
LEAST SQUARES	PLOTTING CONFIDENCE INTERVALS	CIPLOT	1-11-16
LEGENDRE	ASSOCIATED LEGENDRE FUNCTION	ALEGEND	1-2-61
LEGENDRE	LEGENDRE POLYNOMIALS, PN(X)	PN	1-2-60
LIBSTAT	MONITOR USAGE OF LIBRARY ROUTINES	LIBSTAT	2-5-75
LINE	CALCOMP LINE PLOTTING ROUTINE	LINE	2-1-68
LINEAR EQUATIONS	SOLVE A REAL TRIDIAGONAL SYSTEM	TRIEQN	1-8-10
LINEAR EQUATIONS	SOLVE A REAL LINEAR SYSTEM (OUT OF CORE)	OCSOLVE	1-8-04
LINEAR EQUATIONS	SOLVE A REAL LINEAR SYSTEM (IN CORE)	SOLVEQN	1-8-00
LINEAR EQUATIONS	SOLVE A COMPLEX LINEAR SYSTEM	CSOLVEQ	1-8-01
LINEAR SYSTEM	SPARSE SYMMETRIC POS DEF SYSTEM SOLVER	PDSPARL	1-8-23
LINEAR SYSTEM	SOLVE A REAL LINEAR SYSTEM (IN CORE)	SOLVEQN	1-8-00
LINEAR SYSTEM	SOLVE A REAL TRIDIAGONAL SYSTEM	TRIEQN	1-8-10
LINEAR SYSTEM	SOLVE A COMPLEX LINEAR SYSTEM	CSOLVEQ	1-8-01
LINEAR SYSTEM	SOLVE A REAL LINEAR SYSTEM (OUT OF CORE)	OCSOLVE	1-8-04
LOCINT	TABLE LOOK-UP FUNCTION	LOCRL	2-3-01

KEYWORD	TITLE	ROUTINE	
LOCRL	TABLE LOOK-UP FUNCTION	LOCRL	2-3-01
LOGARITHM	LN(GAMMA) FOR REAL ARG	GAMLN	1-2-11
LOGARITHM	GAMMA AND LN(GAMMA) FOR COMPLEX ARG	CGAMMA	1-2-12
LOGLOG	PLOT AND LABEL AXES FOR LOG-LOG PLOTS	LOGLOG	2-1-12
LOGLOGM	PLOT AND LABEL AXES FOR LOG-LOG PLOTS	LOGLOG	2-1-12
LORENTZIAN	GAUSSIAN/LORENTZIAN CONVOLUTION	GLINT	1-3-01
MACHINE DEPENDENCY	CDC MACHINE DEPENDENT CONSTANTS	FLPT	1-1-10
MADD	ADD SUBTRACT MULTIPLY REAL MATRICES	MMULT	1-7-01
MAGNETIC TAPE	DATA MANAGEMENT PACKAGE	DATPAK	2-2-10
MATRIX	DETERMINANT OF A COMPLEX MATRIX	CDET	1-7-04
MATRIX	ADD SUBTRACT MULTIPLY REAL MATRICES	MMULT	1-7-01
MATRIX	ADD SUBTRACT MULTIPLY COMPLEX MATRICES	CMMULT	1-7-02
MATRIX	DETERMINANT OF A REAL MATRIX	DET	1-7-03
MEMORY MANAGEMENT	MEMORY MANAGEMENT (FIXED-LENGTH BLOCKS)	CHNGFX	2-5-32
MEMORY MANAGEMENT	MEMORY MANAGEMENT (EXPANDABLE BLOCKS)	CHNGEX	2-5-33
MEMSCM	RETURN CURRENT FIELD LENGTH -STATIC MODE	MEMSCM	2-5-31
MFID	MAINFRAME IDENTIFICATION ROUTINE	MFID	2-5-42
MINIMIZATION	MINIMIZE SUM OF SQUARES OF DIFF FUNC	NL2SOL	1-11-25
MLSQQ	NONLINEAR LEAST SQUARES FIT TO F(X1,.XK)	MLSQQ	1-11-20
MMULT	ADD SUBTRACT MULTIPLY REAL MATRICES	MMULT	1-7-01
MONTE CARLO	RANDOM NUMBER GENERATOR - EXPONENTIAL	RANEX	1-5-00
MONTE CARLO	RANDOM NUMBER GENERATOR - NORMAL DIST	RANORM	1-5-10
MOVLUV	USE ECS ADDRESSES GREATER THAN 2**17	MOVLUV	2-5-35
MSUB	ADD SUBTRACT MULTIPLY REAL MATRICES	MMULT	1-7-01
MULTIPLY	ADD SUBTRACT MULTIPLY REAL MATRICES	MMULT	1-7-01
MULTIPLY	ADD SUBTRACT MULTIPLY COMPLEX MATRICES	CMMULT	1-7-02
NLSPAR	SOLVE A SPARSE NONLINEAR SYSTEM	NLSPAR	1-10-31
NL2INT	NL2SOL INTERFACE	NL2INT	1-11-24
NL2SNO	MINIMIZE SUM OF SQUARES OF DIFF FUNC	NL2SOL	1-11-25
NL2SOL	MINIMIZE SUM OF SQUARES OF DIFF FUNC	NL2SOL	1-11-25
NONLINEAR	NL2SOL INTERFACE	NL2INT	1-11-24
NONLINEAR	MINIMIZE SUM OF SQUARES OF DIFF FUNC	NL2SOL	1-11-25
NONLINEAR SYSTEM	SOLVE A SPARSE NONLINEAR SYSTEM	NLSPAR	1-10-31
NORMAL DISTRIBUTION	RANDOM NUMBER SEQUENCE - NORMAL DIST	RNORM	1-5-11
NUMBER	PLOT FLOATING POINT NUMBERS	NUMBER	2-1-65
OCSOLVE	SOLVE A REAL LINEAR SYSTEM (OUT OF CORE)	OCSOLVE	1-8-04
OFLOW	TRANSFER CONTROL ON ARITHMETIC ERROR	OFLOW	2-5-10
ORDERB	SORTING DATA WITH TAG ARRAY + ERROR FLAG	ORDERF	2-3-11
ORDERF	SORTING DATA WITH TAG ARRAY + ERROR FLAG	ORDERF	2-3-11
PACK	PACK OR UNPACK BIT STRINGS	PAKUNP	2-4-10
PACK	PACK A SPARSE MATRIX	PACKER	1-7-10
PACK	EVALUATE AND PACK A SPARSE JACOBIAN	JACOB	1-7-11
PACKER	PACK A SPARSE MATRIX	PACKER	1-7-10
PAKUNP	PACK OR UNPACK BIT STRINGS	PAKUNP	2-4-10
PAPER TAPE	PAPER TAPE READING ROUTINE	PTREAD	2-2-30
PAPER TAPE	PAPER TAPE PUNCHING ROUTINE	PTPUN	2-2-31
PCALC	COMPLETE FIT DONE BY RLSTEP	PCALC	1-11-14
PDSPARL	SPARSE SYMMETRIC POS DEF SYSTEM SOLVER	PDSPARL	1-8-23
PI	SINGLE OR DOUBLE PRECISION X*PI	XPI	1-1-00

KEYWORD	TITLE	ROUTINE	
PLODA	PLOT DASHED LINES, OPTIONAL SMOOTHING	PLODAM	2-1-03
PLODAM	PLOT DASHED LINES, OPTIONAL SMOOTHING	PLODAM	2-1-03
PLOGX	BASIC SEMI-LOG X PLOTTING ROUTINE	PLOGX	2-1-01
PLOGXM	BASIC SEMI-LOG X PLOTTING ROUTINE	PLOGX	2-1-01
PLOGXY	BASIC LOG-LOG PLOTTING ROUTINE	PLOGXY	2-1-01
PLOGXYM	BASIC LOG-LOG PLOTTING ROUTINE	PLOGXY	2-1-01
PLOGY	BASIC SEMI-LOG Y PLOTTING ROUTINE	PLOGY	2-1-01
PLOGYM	BASIC SEMI-LOG Y PLOTTING ROUTINE	PLOGY	2-1-01
PLOT	BASIC PLOTTING ROUTINE	PLOT	2-1-00
PLOTC	BASIC PLOTTING ROUTINE	PLOT	2-1-00
PLOTCI	CALCOMP INITIALIZATION ROUTINE	PLOTCI	2-1-60
PLOTI	CALCOMP INITIALIZATION ROUTINE	PLOTCI	2-1-60
PLOTM	BASIC PLOTTING ROUTINE	PLOT	2-1-00
PLOTTING	PLOT AND LABEL AXES FOR SEMI-LOG Y PLOTS	SMLOGY	2-1-11
PLOTTING	PLOT AND LABEL AXES FOR SEMI-LOG X PLOTS	SMLOGX	2-1-11
PLOTTING	CONTOUR SURFACE PLOTTING	CONTOUR	2-1-35
PLOTTING	BASIC PLOTTING ROUTINE	PLOT	2-1-00
PLOTTING	THREE-DIM PERSPECTIVE SURFACE PLOTTING	SPACE	2-1-30
PLOTTING	PLOT DASHED LINES, OPTIONAL SMOOTHING	PLODAM	2-1-03
PLOTTING	PRINTER PLOTTING ROUTINE	PROT	2-1-40
PLOTTING	LINEAR AXES LABELLING ROUTINE	AXLIN	2-1-10
PLOTTING	PLOT AND LABEL AXES FOR LOG-LOG PLOTS	LOGLOG	2-1-12
PLOTTING	BASIC SEMI-LOG Y PLOTTING ROUTINE	PLOGY	2-1-01
PLOTTING	POSITION PLOTTER PEN ON GRAPH PAPER	SPLOT	2-1-50
PLOTTING	LOG-LOG X AND Y AXES LABELLING ROUTINE	AXLGXY	2-1-10
PLOTTING	SIMPLE PLOTTING ROUTINE	SIMPLT	2-1-20
PLOTTING	PLOT TEXT WITH TWO CHARACTER SIZES	VARSIZM	2-1-02
PLOTTING	THREE-DIM SURFACE PLOTTING	PLOT3D	2-1-31
PLOTTING	SEMI-LOG X AXIS LABELLING ROUTINE	AXLGX	2-1-10
PLOTTING	BASIC SEMI-LOG X PLOTTING ROUTINE	PLOGX	2-1-01
PLOTTING	SEMI-LOG Y AXIS LABELLING ROUTINE	AXLGY	2-1-10
PLOTTING	BASIC LOG-LOG PLOTTING ROUTINE	PLOGXY	2-1-01
PLOTTING CALCOMP	CALCOMP LINE PLOTTING ROUTINE	LINE	2-1-68
PLOTTING CALCOMP	RETRIEVE CALCOMP FACTOR AND COORDINATES	WHERE	2-1-62
PLOTTING CALCOMP	CALCOMP SCALING ROUTINE	SCALE	2-1-66
PLOTTING CALCOMP	SET CALCOMP PLOTTING FACTOR	FACTOR	2-1-61
PLOTTING CALCOMP	PLOT TEXT AND SPECIAL SYMBOLS	SYMBOL	2-1-64
PLOTTING CALCOMP	BASIC CALCOMP PLOTTING ROUTINE	PLT	2-1-63
PLOTTING CALCOMP	CALCOMP AXIS PLOTTING ROUTINE	AXIS	2-1-67
PLOTTING CALCOMP	CALCOMP INITIALIZATION ROUTINE	PLOTCI	2-1-60
PLOTTING CALCOMP	PLOT FLOATING POINT NUMBERS	NUMBER	2-1-65
PLOT3D	THREE-DIM SURFACE PLOTTING	PLOT3D	2-1-31
PLOT3DM	THREE-DIM SURFACE PLOTTING	PLOT3D	2-1-31
PLT	BASIC CALCOMP PLOTTING ROUTINE	PLT	2-1-63
PN	LEGENDRE POLYNOMIALS, PN(X)	PN	1-2-60
POLFIT	LEAST SQUARES FIT TO A POLYNOMIAL IN X	POLFIT	1-11-00
POLREG	FIT A GIVEN POLYNOMIAL IN (X1,...XK)	POLREG	1-11-11
POLYNOMIAL	LEGENDRE POLYNOMIALS, PN(X)	PN	1-2-60
POLYNOMIAL	ROOTS OF A QUARTIC EQUATION	QUARTIC	1-10-01
POLYNOMIAL	ROOTS OF A CUBIC EQUATION	CUBIC	1-10-00

KEYWORD	TITLE	ROUTINE	
PRESET	INITIALIZE TO USER-DEFINED PATTERN	PRESET	2-3-30
PRFIT	FIT A POLYNOMIAL OF DEG I IN (X1,...XK)	PRFIT	1-11-10
PRNTSPY	MONITOR PROGRAM EXECUTION	SPYON	2-5-50
PROT	PRINTER PLOTTING ROUTINE	PROT	2-1-40
PSATLW	SATURATION PRESSURE OF LIGHT WATER	PSATLW	1-4-00
PSTATF	CALCULATE STATISTICS FOR REGRESSION	PSTATF	1-11-15
PTPUN	PAPER TAPE PUNCHING ROUTINE	PTPUN	2-2-31
PTREAD	PAPER TAPE READING ROUTINE	PTREAD	2-2-30
QUAD	ADAPTIVE NEWTON-COTES INTEGRATION	QUAD	1-16-03
QUADRATURE	GAUSSIAN DOUBLE INTEGRATION	DUBLINT	1-16-20
QUADRATURE	EXTRAPOLATIVE ROMBERG INTEGRATION	CADRE	1-16-00
QUADRATURE	INTEGRATION OF TABULAR DATA	SPLINT	1-16-30
QUADRATURE	ADAPTIVE SIMPSONS INTEGRATION	COSIMP	1-16-01
QUADRATURE	FILON INTEGRATION	FILON	1-16-10
QUADRATURE	ADAPTIVE NEWTON-COTES INTEGRATION	QUAD	1-16-03
QUADRATURE	GAUSS-HERMITE INTEGRATION	GAHER	1-16-12
QUADRATURE	ADAPTIVE GAUSS INTEGRATION	AGAUSS	1-16-02
QUADRATURE	GAUSS-LAGUERRE INTEGRATION	GALA	1-16-11
QUADRATURE	GAUSS-JACOBI INTEGRATION	GAJAC	1-16-13
QUARTIC	ROOTS OF A QUARTIC EQUATION	QUARTIC	1-10-01
RADCURV	APPROX RADIUS AND CENTRE OF CURVATURE	RADCURV	1-11-30
RADIUS	APPROX RADIUS AND CENTRE OF CURVATURE	RADCURV	1-11-30
RANDOM NUMBERS	RANDOM NUMBER SEQUENCE - NORMAL DIST	RNORM	1-5-11
RANDOM NUMBERS	RANDOM NUMBER SEQUENCE - EXPONENTIAL	REX	1-5-01
RANDOM NUMBERS	RANDOM NUMBER GENERATOR - NORMAL DIST	RANORM	1-5-10
RANDOM NUMBERS	RANDOM NUMBER GENERATOR - EXPONENTIAL	RANEX	1-5-00
RANEX	RANDOM NUMBER GENERATOR - EXPONENTIAL	RANEX	1-5-00
RANORM	RANDOM NUMBER GENERATOR - NORMAL DIST	RANORM	1-5-10
REGRESSION	LEAST SQUARES FIT TO A POLYNOMIAL IN X	POLFIT	1-11-00
REGRESSION	NONLINEAR LEAST SQUARES FIT TO F(X1,.XK)	MLSQQ	1-11-20
REGRESSION	FIT A POLYNOMIAL OF DEG I IN (X1,...XK)	PRFIT	1-11-10
REGRESSION	NL2SOL INTERFACE	NL2INT	1-11-24
REGRESSION	FIT A GIVEN POLYNOMIAL IN (X1,...XK)	POLREG	1-11-11
REGRESSION	MINIMIZE SUM OF SQUARES OF DIFF FUNC	NL2SOL	1-11-25
REGRESSION	PLOTTING CONFIDENCE INTERVALS	CIPLOT	1-11-16
RELESE	RELEASE A FILE	RELESE	2-5-20
RESET	RESET CORE TO LOADER PATTERN	RESET	2-3-31
RESIN	RESONANCE INTEGRAL	RESIN	1-3-00
RESONANCE INTEGRAL	RESONANCE INTEGRAL	RESIN	1-3-00
REX	RANDOM NUMBER SEQUENCE - EXPONENTIAL	REX	1-5-01
RKFINT	INTEGRATION USING RUNGE-KUTTA-FEHLBERG	RKFINT	1-17-10
RKONE	SHORT RUNGE-KUTTA INTEGRATION	RKONE	1-17-00
RNORM	RANDOM NUMBER SEQUENCE - NORMAL DIST	RNORM	1-5-11
ROUTE	ROUTE FILE TO AN OUTPUT QUEUE	ROUTE	2-5-21
SATURATION PRESSURE	SATURATION PRESSURE OF LIGHT WATER	PSATLW	1-4-00
SATURATION TEMP	SATURATION TEMPERATURE OF LIGHT WATER	TSATLW	1-4-01
SBYT	BYTE REPLACEMENT	SBYT	2-4-01
SCALE	CALCOMP SCALING ROUTINE	SCALE	2-1-66
SERIES	SERIES SUMMATION FUNCTIONS	TSUM	1-7-00
SETBAD	SET INDEFINITES FOLLOWING AELIB ERROR	SETBAD	2-5-77

KEYWORD	TITLE	ROUTINE	
TDISTN	STUDENTS T-DISTRIBUTION FUNCTION	TDISTN	1-2-81
THERMAL CONDUCTIVITY	THERMAL CONDUCTIVITY OF LIGHT WATER	TCONLW	1-4-03
TIMEIO	INPUT/OUTPUT TIME USED OR LEFT	TIMEIO	2-5-41
TRACEB	ISSUE TRACEBACK FOR AELIB ERRORS	TRACEB	2-5-74
TRIEQN	SOLVE A REAL TRIDIAGONAL SYSTEM	TRIEQN	1-8-10
TSATLW	SATURATION TEMPERATURE OF LIGHT WATER	TSATLW	1-4-01
TSUM	SERIES SUMMATION FUNCTIONS	TSUM	1-7-00
TURNING POINTS	MINIMUM OR MAXIMUM OF F(X)	STEPMIN	1-10-12
UNPACK	PACK OR UNPACK BIT STRINGS	PAKUNP	2-4-10
USAGE STATISTICS	MONITOR PROGRAM EXECUTION	SPYON	2-5-50
USAGE STATISTICS	MONITOR USAGE OF LIBRARY ROUTINES	LIBSTAT	2-5-75
USERAP	USER ACCOUNTING PARAMETERS	USERAP	2-5-44
VARSIZ	PLOT TEXT WITH TWO CHARACTER SIZES	VARSIZM	2-1-02
VARSIZM	PLOT TEXT WITH TWO CHARACTER SIZES	VARSIZM	2-1-02
VISLW	DYNAMIC VISCOSITY OF LIGHT WATER	VISLW	1-4-04
WCOEF	RACAH OR W-COEFFICIENT	WCOEF	1-3-21
WHERE	RETRIEVE CALCOMP FACTOR AND COORDINATES	WHERE	2-1-62
XCOEF	X-COEFFICIENT	XCOEF	1-3-22
XEXP	X*E TO SINGLE PRECISION	XEXP	1-1-01
XPI	SINGLE OR DOUBLE PRECISION X*PI	XPI	1-1-00
XTIME	CENTRAL PROCESSOR TIME USED OR LEFT	XTIME	2-5-40
YFST	FITTED VALUE STANDARD DEVIATION ESTIM	YFST	1-11-21
ZERO	ZEROES OF F(X) USING AITKEN INTERP.	ZERO	1-10-10
ZEROES	ZEROES OF F(X) USING MUELLERS METHOD	ZEROM	1-10-11
ZEROES	ROOTS OF A CUBIC EQUATION	CUBIC	1-10-00
ZEROES	ROOTS OF A QUARTIC EQUATION	QUARTIC	1-10-01
ZEROES	ZEROES OF F(X) USING AITKEN INTERP.	ZERO	1-10-10
ZEROM	ZEROES OF F(X) USING MUELLERS METHOD	ZEROM	1-10-11

APPENDIX 2

OTHER SUBROUTINE LIBRARIES

IMSLIB

This library is leased from a commercial software organization, International Mathematical and Statistical Libraries Incorporated, Houston, Texas. It contains approximately 400 high quality routines organized in the following chapters.

- Analysis of experimental design data
- Basic statistics
- Categorized data analysis
- Differential equations; quadrature; differentiation
- Eigenanalysis
- Forecasting; econometrics; time series
- Generation and testing of random numbers; goodness of fit
- Interpolation, approximation and smoothing
- Linear algebraic equations
- Mathematical and statistical special functions
- Nonparametric statistics
- Observation structure
- Regression analysis
- Sampling
- Utility functions
- Vector matrix arithmetic
- Zeros and extrema; linear programming

The contents of this library and calling sequence descriptions for all routines are provided in the IMSL Reference Manual Volumes 1, 2, 3 and 4 [4]. Copies of this manual are available to users at most sites. See your site consultant for specific locations.

Routines in this library are provided automatically at load time to NOS/BE programs calling them. To use any IMSL routine on NOS/BE, then, just compile and execute a FORTRAN program which calls this routine according to the documentation specifications.

On the NOS system, explicit control statements are required to access the IMSL routines. On the CRNL CYBER 170 Model 720 use:

```
ATTACH,IMSLIB/UN=LIBRARY.
LIBRARY,IMSLIB.                        with the FTN4 compiler

or

ATTACH,IMSLIB5/UN=LIBRARY.
LIBRARY,IMSLIB5.                       with the FTN5 compiler
```

On the CANDU Operations CYBER 170 Model 720 at Sheridan Park use:

```
ATTACH,IMSLIB/UN=PUBCRNL.                    with the FTN4 compiler
LIBRARY,IMSLIB.
```

or

```
ATTACH,IMSLIB5/UN=PUBCRNL.                   with the FTN5 compiler
LIBRARY,IMSLIB5.
```

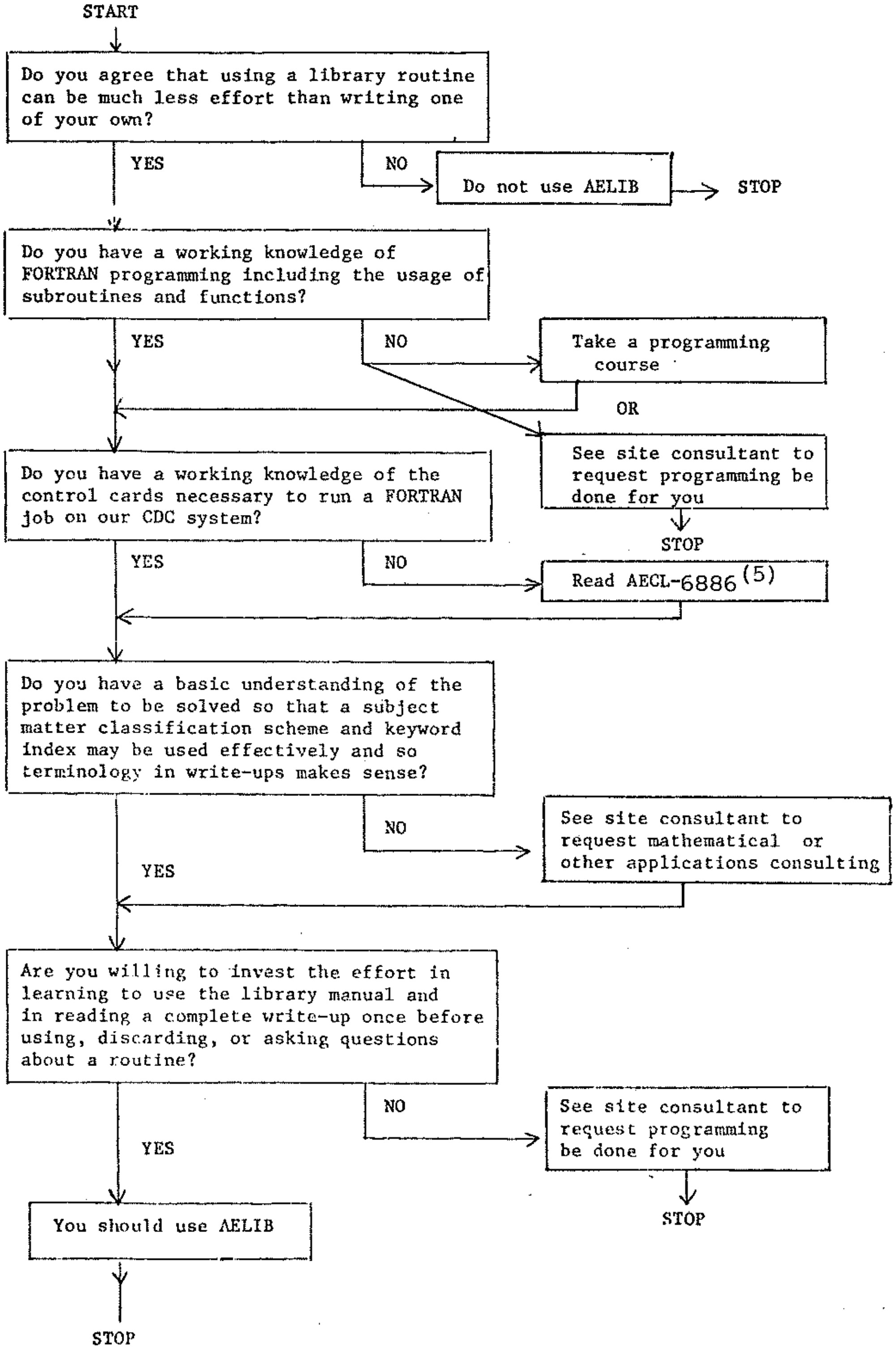
Any queries about this library should be directed to your site consultant.

Harwell Subroutine Library

We have the source code and library manual for the subroutine library from Atomic Energy Research Establishment (AERE), Harwell. These routines are written in IBM FORTRAN and therefore must be modified to run on our system. For information about this library, contact the AELIB librarian via your site consultant.

APPENDIX 3

SHOULD YOU BE USING AELIB?


START
Do you agree that using a library routine can be much less effort than writing one of your own?
YES
NO
Do not use AELIB
STOP
Do you have a working knowledge of FORTRAN programming including the usage of subroutines and functions?
YES
NO
Take a programming course
OR
See site consultant to request programming be done for you
STOP
Do you have a working knowledge of the control cards necessary to run a FORTRAN job on our CDC system?
YES
NO
Read AECL-6886 (5)
Do you have a basic understanding of the problem to be solved so that a subject matter classification scheme and keyword index may be used effectively and so terminology in write-ups makes sense?
NO
See site consultant to request mathematical or other applications consulting
YES
Are you willing to invest the effort in learning to use the library manual and in reading a complete write-up once before using, discarding, or asking questions about a routine?
NO
See site consultant to request programming be done for you
YES
STOP
You should use AELIB
STOP

APPENDIX 4

CROSS-REFERENCE LIST OF SUBROUTINE MODULES USED BY AELIB ROUTINES

Some AELIB "routines" are modular in form and, therefore, actually consist of several subroutines. In these cases, source code for all modules is provided under the deckname of the main routine. If some modules are shared by a group of similar AELIB routines, the source code for the common routines resides with one main routine and is usually clearly referenced by comments in the others. To help users to interpret loader maps, a cross-reference list of these extra subroutine modules has been prepared.

In this list each module (that is not itself an AELIB routine) is listed alphabetically with the name of the AELIB routine using it and the name of the deck in which the source code is stored on the UPDATE program library. For example, the module JAKOB is used by STIFFZ and source code is in the deck STIFFZ.

There is a separate entry for each routine that uses a given module. Therefore, for example, LOGPR1 appears three times in the list, once for each of the routines that call it, PLOGX, PLOGXY and PLOGY.

MODULE	CALLED BY		SOURCE CODE WITH
AECLOS	RELESE	2-5-20	RELESE
AECLOS	ROUTE	2-5-21	RELESE
AEFLFN	DMPFET	2-5-00	RELESE
AEFLFN	ROUTE	2-5-21	RELESE
AEFLFN	RELESE	2-5-20	RELESE
AERETR	RELESE	2-5-20	RELESE
ALRDEL	ALEROF	2-5-73	ALEROF
ALRDEL	ALERON	2-5-73	ALEROF
ALRFND	ALEROF	2-5-73	ALEROF
ALRFND	ALERON	2-5-73	ALEROF
ALRINI	ALEROF	2-5-73	ALEROF
ALRINI	ALERPR	2-5-73	ALEROF
ALRINI	ALERON	2-5-73	ALEROF
ALRTST	ALEROF	2-5-73	ALEROF
ALRTST	ALERPR	2-5-73	ALEROF
ALRTST	ALERON	2-5-73	ALEROF
ASSESS	NL2SOL	1-11-25	NL2SOL
BALANC	EIGRGEN	1-9-10	EIGRGEN
BALBAK	EIGRGEN	1-9-10	EIGRGEN
CALCJ	NL2INT	1-11-24	NL2INT
CALCR	NL2INT	1-11-24	NL2INT
CBABK2	EIGCGEN	1-9-30	EIGCGEN
CBAL	EIGCGEN	1-9-30	EIGCGEN
CHNGECS	CHNGEC	2-5-36	CHNGEC
CMMALC	CHNGFX	2-5-32	CHNGFX
CMMCSF	CHNGEX	2-5-33	CHNGEX
CMMFGR	CHNGFX	2-5-32	CHNGFX
CMMFRE	CHNGFX	2-5-32	CHNGEX
CMMFRE	CHNGEX	2-5-33	CHNGEX
CMMFTB	CHNGFX	2-5-32	CHNGFX
CMMGLF	CHNGEX	2-5-33	CHNGEX
CMMSLF	CHNGEX	2-5-33	CHNGEX
COMHES	EIGCGEN	1-9-30	EIGCGEN
COMLR	EIGCGEN	1-9-30	EIGCGEN
COMLR2	EIGCGEN	1-9-30	EIGCGEN
COSAUX	PLODAM	2-1-03	PLODAM
COSET	STIFFZ	1-17-20	STIFFZ
COVCLC	NL2SOL	1-11-25	NL2SOL
DECB	STIFFZ	1-17-20	STIFFZ
DECOMP	STIFFZ	1-17-20	STIFFZ
DERRES	RADCURV	1-11-30	RADCURV
DFAULT	NL2SOL	1-11-25	NL2SOL
DOTPRD	NL2SOL	1-11-25	NL2SOL
DUMPON	SIMPLT	2-1-20	SIMPLT
DUPDAT	NL2SOL	1-11-25	NL2SOL
ELMHES	EIGRGEN	1-9-10	EIGRGEN
ELTRAN	EIGRGEN	1-9-10	EIGRGEN
FCNINT	PLODAM	2-1-03	PLODAM
FGETJCI	SETCCL	2-5-61	SETCCL
FIGI	EIGRGEN	1-9-10	EIGRGEN

MODULE	CALLED BY		SOURCE CODE WITH
FIGI2	EIGRGEN	1-9-10	EIGRGEN
FSETJCI	SETCCL	2-5-61	SETCCL
GEARZ	STIFFZ	1-17-20	STIFFZ
GQTSTP	NL2SOL	1-11-25	NL2SOL
HQR	EIGRGEN	1-9-10	EIGRGEN
HQR2	EIGRGEN	1-9-10	EIGRGEN
HTRIBK	EIGCHER	1-9-20	EIGCHER
HTRIDI	EIGCHER	1-9-20	EIGCHER
ICMMALF	CHNGFX	2-5-32	CHNGFX
ICMMALF	CHNGEX	2-5-33	CHNGEX
ICMMGEC	CHNGEC	2-5-36	CHNGEC
ICMMGFS	CHNGFX	2-5-32	CHNGEX
ICMMGFS	CHNGEX	2-5-33	CHNGEX
IGETFIT	ROUTE	2-5-21	RELESE
IGETFIT	DMPFET	2-5-00	RELESE
IGETFIT	RELESE	2-5-20	RELESE
IMDCON	NL2SOL	1-11-25	NL2SOL
IMTQL1	EIGRSYM	1-9-00	EIGRSYM
IMTQL1	EIGRGEN	1-9-10	EIGRSYM
IMTQL1	EIGCHER	1-9-20	EIGRSYM
IMTQL2	EIGRSYM	1-9-00	EIGRSYM
IMTQL2	EIGRGEN	1-9-10	EIGRSYM
IMTQL2	EIGCHER	1-9-20	EIGRSYM
ITSMRY	NL2SOL	1-11-25	NL2SOL
JACOBI	GAJAC	1-16-13	GAJAC
JAKOB	STIFFZ	1-17-20	STIFFZ
LINVRT	NL2SOL	1-11-25	NL2SOL
LITVMU	NL2SOL	1-11-25	NL2SOL
LIVMUL	NL2SOL	1-11-25	NL2SOL
LMSTEP	NL2SOL	1-11-25	NL2SOL
LOGPR1	PLOGXY	2-1-01	PLOGX
LOGPR1	PLOGY	2-1-01	PLOGX
LOGPR1	PLOGX	2-1-01	PLOGX
LOGPR10	AXLGX	2-1-10	AXLGX
LOGPR10	AXLIN	2-1-10	AXLGX
LOGPR10	AXLGY	2-1-10	AXLGX
LOGPR11	AXLGY	2-1-10	AXLGX
LOGPR11	AXLGX	2-1-10	AXLGX
LOGPR11	AXLGXY	2-1-10	AXLGX
LOGPR12	PLOGX	2-1-01	PLOGX
LOGPR12	PLOGY	2-1-01	PLOGX
LOGPR12	PLOGXY	2-1-01	PLOGX
LOGPR13	SIMPLT	2-1-20	SIMPLT
LOGPR14	SIMPLT	2-1-20	SIMPLT
LOGPR15	SIMPLT	2-1-20	SIMPLT
LOGPR16	SIMPLT	2-1-20	SIMPLT
LOGPR17	SIMPLT	2-1-20	SIMPLT
LOGPR18	SIMPLT	2-1-20	SIMPLT
LOGPR19	SIMPLT	2-1-20	SIMPLT
LOGPR2	PLOGX	2-1-01	PLOGX

MODULE	CALLED BY		SOURCE CODE WITH
LOGPR2	PLOGXY	2-1-01	PLOGX
LOGPR2	PLOGY	2-1-01	PLOGX
LOGPR20	SIMPLT	2-1-20	SIMPLT
LOGPR21	SIMPLT	2-1-20	SIMPLT
LOGPR3	PLOGXY	2-1-01	PLOGX
LOGPR3	PLOGX	2-1-01	PLOGX
LOGPR3	PLOGY	2-1-01	PLOGX
LOGPR6	PLOGY	2-1-01	PLOGX
LOGPR6	PLOGX	2-1-01	PLOGX
LOGPR6	PLOGXY	2-1-01	PLOGX
LOGPR7	PLOGX	2-1-01	PLOGX
LOGPR7	PLOGXY	2-1-01	PLOGX
LOGPR7	PLOGY	2-1-01	PLOGX
LOGPR8	PLOGX	2-1-01	PLOGX
LOGPR8	PLOGXY	2-1-01	PLOGX
LOGPR8	PLOGY	2-1-01	PLOGX
LOGPR9	AXLGY	2-1-10	AXLGX
LOGPR9	AXLIN	2-1-10	AXLGX
LOGPR9	AXLGXY	2-1-10	AXLGX
LOGPR9	AXLGX	2-1-10	AXLGX
LSQRT	NL2SOL	1-11-25	NL2SOL
LSVMIN	NL2SOL	1-11-25	NL2SOL
LTSQAR	NL2SOL	1-11-25	NL2SOL
MEMECS	CHNGEC	2-5-36	CHNGEC
MSCALE	SPARSE	1-8-20	SPARSE
MSCALE	SPARSEB	1-8-21	SPARSE
NL2ITR	NL2SOL	1-11-25	NL2SOL
NPLIN	NLSPAR	1-10-31	NLSPAR
PARCHK	NL2SOL	1-11-25	NL2SOL
PLFL	PLT	2-1-63	PLT
PLFL	NUMBER	2-1-65	PLT
PLFL	SYMBOL	2-1-64	PLT
PLOTL	PLOGX	2-1-01	SIMPLT
PLOTL	SIMPLT	2-1-20	SIMPLT
QAPPLY	NL2SOL	1-11-25	NL2SOL
QRFACT	NL2SOL	1-11-25	NL2SOL
RELDST	NL2SOL	1-11-25	NL2SOL
RESID	RADCURV	1-11-30	RADCURV
RMDCON	NL2SOL	1-11-25	NL2SOL
RPTMUL	NL2SOL	1-11-25	NL2SOL
RTBLNK	ROUTE	2-5-21	ROUTE
RTCLOS	ROUTE	2-5-21	ROUTE
RTERR	ROUTE	2-5-21	ROUTE
RTINIT	ROUTE	2-5-21	ROUTE
RTSTOR	ROUTE	2-5-21	ROUTE
SLUPDT	NL2SOL	1-11-25	NL2SOL
SLVMOL	NL2SOL	1-11-25	NL2SOL
SOLB	STIFFZ	1-17-20	STIFFZ
SOLVE	STIFFZ	1-17-20	STIFFZ
SORT	EIGRSYM	1-9-00	EIGRSYM

MODULE	CALLED BY		SOURCE CODE WITH
SORT	EIGRGEN	1-9-10	EIGRSYM
SORT	EIGCHER	1-9-20	EIGRSYM
SORT	EIGCGEN	1-9-30	EIGRSYM
STOPX	NL2SOL	1-11-25	NL2SOL
SUBNL	NLSPAR	1-10-31	NLSPAR
SUBPLOT	CIPLOT	1-11-16	CIPLOT
SUMSQ	NLSPAR	1-10-31	NLSPAR
TRED1	EIGRSYM	1-9-00	EIGRSYM
TRED2	EIGRSYM	1-9-00	EIGRSYM
TRIDIAG	EIGRGEN	1-9-10	EIGRSYM
TRIDIAG	EIGRSYM	1-9-00	EIGRSYM
VAXPY	NL2SOL	1-11-25	NL2SOL
VB06A	SMOOTH	1-12-00	SMOOTH
VCOPY	NL2SOL	1-11-25	NL2SOL
VSCOPY	NL2SOL	1-11-25	NL2SOL
V2NORM	NL2SOL	1-11-25	NL2SOL

APPENDIX 5

ERROR PROCESSING FOR AELIB ROUTINES

Error Processing Philosophy

Almost all AELIB routines do some type of error checking. This may be as simple as checking the range of parameter values or as complex as identifying the breakdown of the method being used. Before 15 December 1975, no consistent error processing procedure was followed. Since that time, however, a comprehensive error processing philosophy has been adopted.

(a) For mathematical routines and utility routines not performing input or output functions, the user is given considerable flexibility in dealing with error conditions while ensuring that meaningless results are not used in subsequent calculations. This is done as follows:

1. Flag variables are frequently used to report errors to the user's program and to distinguish among several possible errors.

2. An error is fatal if no further processing can be done by the subroutine. In this case, all output parameters are set to their best value. If no value is meaningful, output parameters are usually set to indefinites to prevent further computation using them. Control is returned to the user's program following a fatal error but the job may be terminated as in (b) if the subroutine is re-entered following a fatal error.

3. An error is non-fatal if recovery from it is possible and is attempted by the subroutine. Control is returned to the user's program following a non-fatal error.

4. Diagnostic messages are always issued to the file OUTPUT in the case of fatal errors. For non-fatal errors, messages are usually issued, but the error flag may be used in some routines to avoid excessive output. Limited user control over messages is provided by AEFREQ (for those routines calling AELERR). More general user control is provided with the ALEROF and ALERON routines (for use with those routines calling ALERPR).

(b) For input/output routines, such as PLOT, PTREAD, and PTPUN, error processing is the same as above except that control is not returned to the user's program following a fatal error. Instead, the job is terminated. Also, there is no user control over message printing. This is consistent with CDC's handling of input/output errors.

Error Message Processing

The mechanism for issuing error messages from AELIB routines has evolved rather quickly. Prior to December 1975, messages were issued by PRINT statements. In December 1975 a local system dependent error message processor, AELERR, was introduced primarily to standardize the form of messages to provide traceback following errors, and to give user control over messages (using AEFREQ). More recently, coding and testing difficulties as well as code portability concerns have resulted in a more moderate error message mechanism, namely PRINT statements with standardized formats plus calls to a separate traceback subroutine, TRACEB. User control is provided by interface routines, ALERPR, ALEROF, and ALERON.

Error Processing for AELIB Subroutines used outside of AELIB

Routines taken from AELIB are sometimes used from private libraries on our system or at other computing installations. We call these external versions of AELIB routines. The error processing philosophy described above is fairly general and, except for the use of indefinites to prevent the use of meaningless numbers, need not be changed for these external versions.

Error messages that are issued by PRINT statements do not impose any additional portability problems, provided the code is being transferred to another CDC machine. Note that we have not claimed that library routines are portable, in fact almost all of Part B, Section 2, is not. We do claim that issuing PRINT statements does not make code less portable. In addition the traceback and user control routines are modular and can be easily removed if not suitable for another system.

The AELERR/AEFREQ error message handling system, however, is designed specifically for use by routines on our system libraries. External versions of AELIB routines will not run with this message handling system. One way of making routines portable is to replace all calls to AELERR by print statements, using messages specified in AELERTX. For many error messages, however, this process can be tedious. A version of AELERR for use with external copies of AELIB routines can be retrieved from the source tape by including the UPDATE directive *DEFINE EXTERNAL along with *C AELERR in the run that requests the source code. This routine prints message numbers only, so a listing of the error message text (stored as the deck AELERTX on the source tape) should accompany the source code to another installation.

APPENDIX 6

COMPARISON OF RESULTS IN A DUAL COMPILER, DUAL OPERATING SYSTEM ENVIRONMENT

Compiler Generated Differences

Numerical results generated by programs and subroutines compiled using the FTN4 compiler may be different from those compiled using the FTN5 compiler. For different compilers, or different versions of the same compiler, there are usually differences in the object code produced for a given program or subroutine. In particular, the order of arithmetic operations can be different and the effect of this change can be small or large, depending on the type of calculations being done. In most cases, these differences should be limited to the last one or two significant digits in the calculated result. If this is not the case and differences are larger than expected, the source of the discrepancy should be determined by tracking down the section of code in which they are produced.

The 1982/83 compilation and testing of AELIB5 has yielded small but acceptable differences from the FTN4 versions in many of the routines. Testing of the eigenanalysis routines turned up a different set of eigenvectors than was expected. On investigation, it was found that both this set and the set produced by AELIB (i.e., the FTN4 version) are accurate eigensolutions and that small round-off differences in the calculations caused the different sets to be generated.

System Effects

Different results can be produced for other reasons than a change in the object code produced by the compiler. These reasons include

- differences in the round options selected for arithmetic operations,
- differences in intrinsic or utility subprograms associated with the compiler,
- differences in operating system philosophy and system parameters.

We encountered some differences in subprograms during compilation and testing of AELIB5 on NOS/BE but do not expect major problems with system effects because the operating system environment is unchanged and the subprograms are standardized. However, there are several system differences between the NOS and NOS/BE systems that can cause different results for programs run under the same compiler on NOS and NOS/BE. These differences are as follows:

(1) Underflow

The condition called underflow is handled differently by the 6600 and the 175 (see p. 2 of the CRNL Modification inserted after p. vi of the NOS/BE Reference Manual, CDC Pub. No. 60493800). The 720 hardware handles underflow in the same way as the 6600; therefore, programs encountering this situation may differ when run on the 175 from when they are run on the 720.

For discussion of underflow and how to avoid it, see the Newsletter, Vol. 6, No. 1, p. 6, and Vol. 8, No. 6, p. 6. (The more general problem of ill-conditioning is discussed in the document MCTD-13, Ill-Conditioning in FORTRAN Programs, available as described in the Newsletter heading.)

(2) Target Machine Differences

The CYBER 175 and 6600 have multiple functional units, allowing over-lapping of some operations, while the 720 CPU is a serial processor, executing only one operation at a time. This difference is reflected in CDC's FORTRAN compilers, which optimize code differently according to whether it is to be executed on a parallel or serial CPU. FTN4 on the CRNL 720 has been installed so that it uses the optimization appropriate for the 175/6600; FTN4 on the Sheridan Park 720 uses the serial machine optimization. As a result, there will be instruction sequences that are executed in a different order on the Sheridan Park 720 than on the other three machines in the "network", and some discrepancies in computed results are, therefore, possible.

(3) Release Level Differences

The NOS/BE and NOS systems will not necessarily always be at the same level. While we do not expect major problems due to differences in either the compilers or the execution-time library, such differences could conceivably exist. Where possible, our policy is to eliminate these differences by bringing the NOS and NOS/BE systems to the same level and using the same system defaults. Where such standardization is not possible it may be awkward to validate results for programs that are being executed on more than one system.

ACKNOWLEDGEMENTS

In addition to the contributors cited, the editors would like to thank the users of AELIB, including the Mathematics and Computation Branch staff, for their confidence in this library, for their never-ending comments and suggestions on its content and documentation, and for their patience during the preparation of this manual. The editors would also like to thank Fay Barnard, Joan Vaudry and Gloria Bateman for their tireless typing efforts over the years.

REFERENCES

[1] E.A. Okazaki (Editor), Computing Centre Newsletter, Monthly Informal Publication of Atomic Energy of Canada Limited.

[2] FORTRAN Extended Version 4 Reference Manual, Revision E, Publication Number 60497800, Control Data Corp., 1979.

[3] FORTRAN Extended Version 5 Reference Manual, Revision F, Publication Number 60481300, Control Data Corp., 1982.

[4] The IMSL Library, Vols. 1, 2, 3 and 4, Library 3, Edition 9, International Mathematical and Statistical Libraries, Inc., 1982.

[5] J.M. Blair, L.E. Evans, E.A. Okazaki and G.N. Williams, Introduction to Computing on the CYBER 170/6600 System, Atomic Energy of Canada Limited, report AECL-6886, Rev. A, 1981 April.

PART B - Documentation for AELIB Routines

1- 1 MATHEMATICAL AND PHYSICAL CONSTANTS

1-1-00	XPI	SINGLE OR DOUBLE PRECISION X*PI
1-1-01	XEXP	X*E TO SINGLE PRECISION
1-1-02	EULER	X*EULERS CONSTANT TO SINGLE PRECISION
1-1-10	FLPT	CDC MACHINE DEPENDENT CONSTANTS

TITLE — Computation of the Product, $x\pi$ to Single or Double Precision.

ENTRY — R = XPI(X)

X real input variable, the multiplier for π.

If x is exactly representable internally in single precision, for example, 1.0, 0.5, or 0.25, the result, $x\pi$ is correct to double precision. If not, it is only correct to single precision.

To obtain the double precision result, the calling program must declare XPI and R to be double precision. If it does not, the result is truncated to single precision, not rounded.

EXIT — Control is returned to the calling program and the value of $x\pi$ is stored in R.

ACCURACY — $\sim$14 significant figures single precision
$\sim$29 significant figures double precision

PROGRAM EXAMPLE — The sequence

```
     X=4.0
     R=XPI(X)
```

calculates and stores 4π to single precision machine accuracy in R.

AUTHOR — J.M. Blair DATE March 1974

TITLE Evaluation of the Product x*e, where e = exp(1) = 2.71828 18284 59045.

ENTRY R=XEXP(X)

X real input variable.

R real output variable in which the product x*e is stored.

EXIT This function computes the product x*e and returns control to the calling routine. No error checking is done.

ACCURACY Correct to single precision.

SPEED 1. x 10^{-5} (6600, May 1978).

AUTHOR B.E. Purcell DATE July 1978

TITLE Evaluation of the Function x*γ where Euler's constant, γ = 0.57721 56649 01533.

ENTRY R=EULER(X)

X real input variable.

R real output variable in which the product x*γ is stored.

EXIT This function computes the product x*(Euler's constant) and returns control to the calling routine. No error checking is done.

ACCURACY Correct to single precision.

SPEED 1.0×10^{-5} s (6600, May 1978)

AUTHOR B.E. Purcell DATE July 1978

TITLE Evaluation of Machine Dependent Constants for the CDC 6600 or Cyber 170.

ENTRY R=FLPT(I)

I integer input variable specifying the constant required as follows:

I=1 largest positive floating point number ($\simeq 1.27*10^{322}$).

I=2 smallest positive normalized floating point number ($\simeq 3.13*10^{-294}$).

I=3 smallest positive floating point number which will significantly add to 1.0 ($\simeq 3.55*10^{-15}$). (Note that this constant has been evaluated with the round-off option on. Without round-off, this constant should be $\simeq 7.11*10^{-15}$).

I=4 largest positive integer such that all positive integers less than or equal to it can be converted exactly to a floating point number ($=2^{48}-1$).

R real output variable in which the machine dependent constant is stored.

EXITS Normal Exit

The value of FLPT is set (depending on the value of I) and control is returned to the calling routine.

Error Exit

If the value of I is less than 1 or greater than 4, the following message is printed

```
***ILLEGAL VALUE FOR I.  I MUST BE 1, 2, 3 OR 4
```

and control is returned to the calling routine.

ACCURACY Correct to single precision.

SPEED Execution times on the CDC 6600 (May 1978) are as follows:

FLPT(1) requires 1.7 x 10^{-5} s
FLPT(2) requires 1.8 x 10^{-5} s
FLPT(3) requires 1.9 x 10^{-5} s
FLPT(4) requires 1.7 x 10^{-5} s

AUTHOR B.E. Purcell DATE May 1978

1- 2 SPECIAL MATHEMATICAL AND STATISTICAL FUNCTIONS

00-04 EXPONENTIAL INTEGRAL

1-2-00	EI	EXPONENTIAL INTEGRAL
1-2-01	EN	EXPONENTIAL INTEGRAL

05-09 ERROR FUNCTION

1-2-06	AINERF	AINERFC INVERSE ERROR FUNCTION

10-19 GAMMA FUNCTION

1-2-10	GAMMA	GAMMA FUNCTION FOR REAL ARGUMENT
1-2-11	GAMLN	LN(GAMMA) FOR REAL ARG
1-2-12	CGAMMA	CGAMLN GAMMA AND LN(GAMMA) FOR COMPLEX ARG
1-2-13	GAMIN	INCOMPLETE GAMMA FUNCTION

20-29 ELLIPTIC INTEGRALS

1-2-20	ELK	ELKCOMP COMPLETE ELLIPTIC INTEGRAL OF 1ST KIND
1-2-21	ELE	ELECOMP COMPLETE ELLIPTIC INTEGRAL OF 2ND KIND
1-2-22	CEL3	EL3COMP COMPLETE ELLIPTIC INTEGRAL OF 3RD KIND
1-2-23	EINCO	INCOMPLETE ELLIPTIC INTEGRALS 1 AND 2

30-49 BESSEL FUNCTIONS AND AIRY FUNCTIONS

1-2-30	AJYBES	BESSEL FUNCTIONS J0,J1,Y0,Y1, REAL ARG
1-2-34	AI0BES	MODIFIED BESSEL FUNCTION I0, REAL ARG
1-2-35	AI1BES	MODIFIED BESSEL FUNCTION I1, REAL ARG
1-2-36	AK0BES	MODIFIED BESSEL FUNCTION K0, REAL ARG
1-2-37	AK1BES	MODIFIED BESSEL FUNCTION K1, REAL ARG
1-2-38	BESGEN	BESSEL FUNCTIONS, INTGR ORDER, REAL ARG
1-2-39	JBESS	J BESSEL FUNCTION OF REAL ORDER AND ARG
1-2-40	IBESS	I BESSEL FUNCTION OF REAL ORDER AND ARG
1-2-41	COMBES	BESSEL FUNCTIONS IN,JN FOR COMPLEX ARG
1-2-42	JBESCPX	BESSEL FUNCTION JP(Z),REAL P, COMPLEX Z
1-2-43	JAIRY	AIRY FUNCTION AND DERIVATIVE, REAL ARG
1-2-44	CBESY0	BESSEL FUNCTION Y0 FOR COMPLEX ARG
1-2-45	CBESY1	BESSEL FUNCTION Y1 FOR COMPLEX ARG

50-59 INTEGRALS OF BESSEL FUNCTIONS

1-2-50	AKI1	BICKLEY FUNCTION KI1
1-2-51	AKI3	BICKLEY FUNCTION KI3
1-2-52	KIN	BICKLEY FUNCTION KIN

60-69 LEGENDRE POLYNOMIALS, ASSOCIATED LEGENDRE FUNCTION

1-2-60	PN	LEGENDRE POLYNOMIALS, PN(X)
1-2-61	ALEGEND	ASSOCIATED LEGENDRE FUNCTION

80-89 OTHER STATISTICAL FUNCTIONS

1-2-80	CHISQ	CHI-SQUARE PROBABILITY FUNCTION
1-2-81	TDISTN	STUDENTS T-DISTRIBUTION FUNCTION

90-99 INVERSE HYPERBOLIC TRIGONOMETRIC FUNCTIONS

1-2-90	ARSINH	INVERSE HYPERBOLIC SINE

1-2 SPECIAL MATHEMATICAL AND STATISTICAL FUNCTIONS

The notation and terminology in this section conform with the usage in [1].

There is some overlap between this AELIB section and chapter M of the IMSL Library [2]. IMSL contains more statistics functions, while AELIB contains more applied mathematics functions. A programmer wishing to compute a special function should first consult the two chapters mentioned. If the required function is not available, he should apply simple mathematical transformations to try to reduce the function to a combination of known functions. For example, the integral

$$\int_{x_1}^{x_2} e^{-1/x}\, dx,$$

under the change of variable t = 1/x becomes

$$\int_{1/x_2}^{1/x_1} \frac{e^{-t}}{t^2}\, dt$$

$$= \int_{1/x_2}^{\infty} \frac{e^{-t}}{t^2}\, dt - \int_{1/x_1}^{\infty} \frac{e^{-t}}{t^2}\, dt$$

$$= x_2 \int_1^{\infty} \frac{e^{-u/x_2}}{u^2}\, du - x_1 \int_1^{\infty} \frac{e^{-u/x_1}}{u^2}\, du$$

(by the substitution t = u/x)

$$= x_2\, E_2\, (1/x_2) - x_1 E_2 (1/x_1)$$

where E_2 is the exponential integral EN. Mathematics and Computation Branch staff will give advice and assist in this task. If the function is not available, and the programmer wishes to have it added to AELIB, he should forward the request to Mathematics and Computation Branch, and we will try to provide an efficient subprogram.

Most of the functions of a single real variable in this chapter are approximated by rational minimax approximations [3], possibly combined with elementary functions. A rational approximation is a function of the form P(x)/Q(x), where P and Q are polynomials. In a minimax approximation the coefficients in P and Q are adjusted to make the error in the approximation "equal ripple"; that is to make the error oscillate between equal positive and negative extreme values over the range of approximation.

1-2 SPECIAL MATHEMATICAL AND STATISTICAL FUNCTIONS

Functions of more than one variable and functions of a complex variable are calculated by a variety of techniques including power and asymptotic series expansions, recursion, and continued fraction expansions.

J.M. Blair
May 1981

References

[1] M. Abramowitz and I.A. Stegun (Ed.), Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, Washington, DC 20402, 1964.

[2] IMSL Library Reference Manual.

[3] J.F. Hart, et al., Computer Approximations, Wiley, New York, 1968.

TITLE Exponential Integral

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

where $-\!\!\!\!\!\!\int$ denotes the Cauchy principal value of the integral.

ENTRY R = EI(X)

X real input variable

EXIT This function returns to the calling program, storing the real value of EI(X) in R. If $x \leq 0$, a message is printed and EI(X) is set to $-\infty$.

ACCURACY Accurate to at least 13 significant figures.

SPEED	$0<x<6$	$\sim$140 µs	(6600, May 1973)
	$x \geq 6$	$\sim$ 90 µs	(6600, May 1973)

AUTHORS J.M. Blair and S. Bérubé DATE May 1973

TITLE Exponential Integral

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n}\, dt$$

ENTRY R = EN(X,N)

X real input variable
N integer input variable $0 \le N \le 200$

EXIT This function returns the real value of $E_n(x)$.

If X<0, or N<0, an error message is printed and EN(X,N) is set to 0.

If X = 0 and N = 0 or 1, a message is printed and an infinite result is returned.

If N > 200, a message is printed and an infinite result is returned.

ACCURACY EN(X,N) is accurate to at least 13 digits for all X and N.

SPEED

X	N	TIME (6600 September 1970)		
All	0	$\sim$80 μs		
All	1	$\sim$140 μs		
$x<5$	All	$\sim$130 N μs		
$x>5$	$n<5$	Variable $E_3(10)$	$\sim$2	ms
		$E_3(40)$	$\sim$1.6	ms
		$E_3(100)$	$\sim$2.2	ms
$x>5$	$5<n<80$	$\sim \frac{4}{\sqrt[3]{x}} + .013\lvert X-N\rvert$ ms		
$x>5$	$n>80$	$\sim .9 + .013\lvert X-N\rvert$ ms		

AUTHORS J.M. Blair and J.H. Schmidt DATE September 1970

AINERF
AINERFC

TITLE Inverse Error Function inverf(x), $0 \le x < 1$

If y = inverf(x), then x = erf(y), where erf(y) denotes the error function.

ENTRY R = AINERF(X)

X real, input variable, for arguments not close to 1.0.

R = AINERFC(Y)

Y real input variable, where Y = -ALOG(1.0-X). If x is close to 1, this entry point should be used to avoid cancellation errors in calculating 1 - x.

EXIT For $0 \le x < 1$ the function returns the real value of inverf(x). For any other x, the function returns an infinite result and issues an appropriate message.

ACCURACY 13 significant figures

SPEED 73 - 109 μs (6600, December 1972)

AUTHOR J.M. Blair DATE December 1972

TITLE Gamma Function $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$

ENTRY R = GAMMA(X)

X real, input variable.

COMMON BLOCKS USED AELERCM

EXIT This function returns the real value of $\Gamma(x)$. If x is a negative integer, 0, or outside the range [-164.1, 177.8], the function value $\Gamma(x)$ is set to infinity and an error message is issued.

ACCURACY ∿14 digits for most cases. However, if x is very close to a negative integer, a loss of significant figures occurs during the calculation.

AUTHORS J.M. Blair and C.J. Johnson DATE January 1969

REVISED B.E. Purcell DATE July 1977

TITLE Natural Logarithm of the Gamma Function, $\ell n\,\Gamma(x)$, $x > 0$

ENTRY R = GAMLN(X)

X real input variable

EXIT The function returns the real value of the natural logarithm of the Gamma function. If $X \leq 0$, an error message is printed, the function is set to infinity, and control is returned to the calling program.

ACCURACY $\sim$13 significant figures

SPEED $\sim$75 μs (6600, March 1975)

AUTHORS D.E. Amos and S.L. Daniel, Sandia Laboratories, Albuquerque, New Mexico, Implemented at CRNL by A. Perreault and J.M. Blair.

DATE March 1975.

TITLE

Gamma Function $\Gamma(z) = \int_o^\infty t^{z-1} e^{-t} dt$
and $\ell n\ \Gamma(z)$, z complex.

ENTRY

C = CGAMMA(Z) for $\Gamma(z)$
C = CGAMLN(Z) for $\ell n\Gamma(z)$
Z complex input variable

The calling program must declare C, CGAMMA, and CGAMLN to be COMPLEX, if it uses them.

EXIT

This function returns the complex value of $\Gamma(z)$ or $\ell n\Gamma(z)$.

If $z = x + iy$, the limits for x and y are:

(1) For $x > 0$ and $|y| < 194$:
$x \leq 177.8 + 0.0057\ |y| + 0.0043\ y^2$

(2) The smallest z that can be used is
$z = -118. \pm 118i$

(3) For y=0. and x a nonpositive integer, an error message is printed and the function is set to an infinite value.

(4) For values outside the ranges given in (1) and (2), the function returns an infinite value for $\Gamma(z)$ but no error message is printed.

ACCURACY

For $0. \leq x \leq 10.$ and $|y| < 10.$, the results are accurate to at least 12 significant digits except for a region very near the origin. No tabular checks are available outside this range; however, for large values of x or y, Stirling's asymptotic series should give even greater accuracy.

If $\delta(=10^{-n}$, n integer $\geq 3)$ is the distance in the complex plane from z to the nearest pole, the minimum number of significant figures = 14-n.

SPEED

x	y	Time(ms) (6600, June 1976)
$x<-10$	0	.55
$\|x\|<1$	0	1.5
$1<\|x\|<10$	0	$1.4-.1\|x\|$
$x>10$	0	.33
0	$\|y\|\leq 12$	1.5
0	$\|y\|>12$	$.12\|y\|$
$\|x\|<10$	$\|y\|<10$	1.5
$\|x\|>10$	$\|y\|<\|x\|$	.4
$\|x\|>10$	$\|y\|>\|x\|$	$.4+.12(\|y\|-\|x\|)$

AUTHOR J.H. Schmidt DATE May 1971

REVISED J.M. Blair June 1976

TITLE Incomplete Gamma Function

$$\Gamma(a,x) = \int_x^\infty e^{-t} t^{a-1} dt \quad \text{where } x \geq 0,\ a \geq 0.$$

ENTRY R = GAMIN(A,X)

A real input variable
X real input variable

ROUTINES CALLED GAMMA from AELIB

EXIT This function returns the real value of $\Gamma(a,x)$ unless the following errors occur: If $X < 0$, or $A < 0$, or $X = A = 0$, an error message is printed and the function value is set to 0.

ACCURACY The error is not greater than 1.5 in the 13th figure.

SPEED .3 ms for $x > a$
.7 ms for $x \leq a$

AUTHOR J.L. Barton DATE April 1971

TITLE Complete Elliptic Integral of the First Kind

$$K(m) = \int_0^{\pi/2} (1-m \sin^2 \theta)^{-\frac{1}{2}} d\theta \quad 0 \leq m < 1$$

ENTRIES R = ELK(M) computes K(m)
R = ELKCOMP(M1) computes K(m), where $m_1 = 1-m$.
This entry point should be used if m is close to 1 and m_1 is known accurately.

M, M1 real input variables

EXIT This function returns the value of the required elliptic integral. If $m<0$ or $m \geq 1$, the value of the function is set equal to zero and an error message is printed.

ACCURACY The accuracy is about 14 digits for most cases. If m is close to one, accuracy is increased by using entry point ELKCOMP.

SPEED Approximately 1.20 ms for all values (6600, June 1970)

AUTHOR S.L. Likeness DATE June 1970

TITLE Complete Elliptic Integral of the Second Kind

$$E(m) = \int_0^{\pi/2} (1-m \sin^2 \theta)^{\frac{1}{2}} d\theta \quad 0 \le m < 1$$

ENTRIES R = ELE(M) computes E(m)

R = ELECOMP(M1) computes E(m), where $m_1 = 1-m$. This entry point should be used if m is close to 1 and m_1 is known accurately.

M, M1 real input variables

EXIT This function returns the value of the required elliptic integral. If $m < 0$ or $m \geq 1$, the value of the function is set to zero and an error message is printed.

ACCURACY The accuracy is about 14 digits for most cases. If m is close to one, accuracy is increased by using entry point ELECOMP.

SPEED Approximately 1.183 ms for all values (6600, June 1970)

AUTHOR S.L. Likeness DATE June 1970

TITLE Complete Elliptic Integral of the Third Kind

$$\Pi(n,m) = \int_0^{\pi/2} (1-n \sin^2 \theta)^{-1} (1-m \sin^2 \theta)^{-\frac{1}{2}} d\theta$$

$$0 < m < 1$$

$$n < 1$$

ENTRIES R = CEL3(N,M,EPS) computes $\pi(n,m)$
R = EL3COMP(N,M1,EPS) computes $\pi(n,m)$, where $m_1 = 1-m$. This entry point should be used if m is close to 1 and m_1 is known accurately.

N	real input variable
M	real, input variable
M1	real, input variable M1 = 1-M
EPS	real, input variable, the relative error criterion required by EINCO to calculate the incomplete elliptic integrals which are used in the calculation of this function.

ROUTINES CALLED ELE/ELECOMP, ELK/ELKCOMP, EINCO from AELIB

EXIT The function returns a real value for the complete elliptic integral of the third kind. The value of the function is set to infinity and error messages are printed for the following illegal arguments:

i) $n \geq 1.0$

ii) $m \geq 1.0$

iii) $m < 0$

ACCURACY The result will be within EPS of the true value.

SPEED Depends on EPS.
∿1 ms with $\varepsilon = 10^{-14}$ (6600, June 1974)
∿.8 ms with $\varepsilon = 10^{-09}$ (6600, June 1974)

AUTHOR L. Yamazaki DATE June 1974

TITLE

Incomplete Elliptic Integrals of the First and Second Kinds.

First Kind:

$$F(\phi,m) = \int_0^{\phi} (1-m \sin^2 \theta)^{-\frac{1}{2}} d\theta$$

Second Kind:

$$E(\phi,m) = \int_0^{\phi} (1-m \sin^2 \theta)^{\frac{1}{2}} d\theta$$

$$\left.\begin{array}{l} \\ \end{array}\right\} \begin{array}{l} 0 \leq m \leq 1 \\ 0 \leq \phi \leq \pi/2 \end{array}$$

ENTRY

CALL EINCO(M1,PHI,CPHI,IFLAG,F,E,EPS)

M1 real input variable, M1=1-M is used as the argument instead of M.

PHI real input variable, the upper limit of integration.

CPHI real input variable, COS(PHI)

IFLAG integer input variable, control on the parameters being supplied.

IFLAG=1	PHI only
IFLAG=2	CPHI only
IFLAG=3	both PHI and CPHI supplied

F real output variable to return the incomplete integral of the first kind.

E real output variable to return the incomplete integral of the second kind.

EPS real input variable, the relative error criterion.

ROUTINES CALLED

ELECOMP, ELKCOMP, XPI from AELIB

EXIT

The subroutine returns with real results in F and E. Error messages are printed for the following illegal arguments:

i)	if $M1 > 1$	$F=E=\pi/2$.
ii)	if $M1 < 0$	E=1, F set to infinity.
iii)	if $\phi > \pi/2$ or $\cos(\phi) < 0$	F and E are set to the values of the complete elliptic integrals.
iv)	if $\phi < 0$ or $\cos(\phi) > 1$	$F=E=0$.

ACCURACY

The result will be within EPS of the true value.

SPEED

Depends on EPS as well as internal boundary conditions which determine which of two different methods within the routine is used.

Average times are:

1.15 ms with EPS = 10^{-14} (6600, June 1974)
.68 ms with EPS = 10^{-9} (6600, June 1974)

AUTHOR

L. Yamazaki DATE June 1974

TITLE Bessel Functions $J_0(x)$, $J_1(x)$, $Y_0(x)$, $Y_1(x)$

ENTRY

```
R = AJYBES(X,1.,0) for J0(x)
R = AJYBES(X,2.,0) for J1(x)
R = AJYBES(X,3.,0) for Y0(x)
R = AJYBES(X,4.,0) for Y1(x)

R = AJYBES(X,P,1) for all four functions.
```

X real input variable

P(4) real output array to return all four Bessel function values, if required.

(Note: When only one value is to be returned, the second parameter in the function call must be given as a real number.)

EXIT The function returns the value of the requested Bessel function. When all functions are requested, J_0 is stored at P(1), J_1 at P(2), Y_0 at P(3) and Y_1 at P(4), with J_0 also returned in R.

STORAGE 1500_8

ACCURACY 13 significant figures except near zeros of the functions where precision decreases as a zero is approached.

SPEED

	$x<8$	$x>8$	
$J_0(x)$	.32 ms	.53 ms	6600 April 1969
$J_1(x)$	.31 ms	.53 ms	
$Y_0(x)$	.62 ms	.54 ms	
$Y_1(x)$	.62 ms	.53 ms	
All four	1.22 ms	1.22 ms	

AUTHOR A.E. Russon DATE April 1969

TITLE Modified Bessel Function $I_0(x)$

ENTRY R = AIOBES(X)

X - real input variable

ROUTINES CALLED LEGVAR from FORTRAN

EXIT Normal Exit: If x is a legal argument, the function returns the value of the Bessel Function $I_0(x)$ to the calling routine.

Error Exit: If x is indefinite or infinite, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is issued, and control is returned to the calling routine.

ACCURACY $I_0(x)$ is accurate to approximately 14 digits.

SPEED For $x < 15$, the function requires 58 µs. (6600, July 1975)
For $x \geq 15$, the function requires 89 µs. (6600, July 1975)

AUTHOR B.E. Purcell DATE July 1975

TITLE Modified Bessel Function $I_1(x)$

ENTRY R = AI1BES(X)

X - real input variable

ROUTINES CALLED LEGVAR from FORTRAN

EXIT Normal Exit: If x is a legal argument, the function returns the value of the Bessel Function $I_1(x)$ to the calling routine.

Error Exit: If x is indefinite or infinite, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is issued, and control is returned to the calling routine.

ACCURACY $I_1(x)$ is accurate to approximately 14 digits.

SPEED For $x < 15$, the function requires 58 μs. (6600, July 1975)
For $x \geq 15$, the function requires 88 μs. (6600, July 1975)

AUTHOR B.E. Purcell DATE July 1975

TITLE Modified Bessel Function $K_0(x)$.

ENTRY R = AKOBES(X)

X - real input variable, $x > 0.0$.

ROUTINES CALLED LEGVAR from FORTRAN

EXIT Normal Exit: If x is a legal argument, the function returns the value of the Bessel Function $K_0(x)$ to the calling routine.

Error Exit: If x is indefinite, infinite, negative, or zero, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is issued, and control is returned to the calling routine.

ACCURACY $K_0(x)$ is accurate to approximately 14 digits.

SPEED For $0 < x \le 1.0$, the function requires 80 μs. (6600, July 1975)
For $x > 1.0$, the function requires 89 μs. (6600, July 1975)

AUTHOR B.E. Purcell DATE July 1975

TITLE Modified Bessel Function $K_1(x)$

ENTRY R = AK1BES(X)

X - real input variable, $x > 0.0$

ROUTINES CALLED LEGVAR from FORTRAN

EXIT Normal Exit: if x is a legal argument, the function returns the value of the Bessel Function $K_1(x)$ to the calling routine.

Error Exit: If x is indefinite, infinite, negative, or zero, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is issued, and control is returned to the calling program.

ACCURACY $K_1(x)$ is accurate to approximately 14 digits.

SPEED For $0 < x \leq 1.0$, the function requires 83 μs. (6600, July 1975)
For $x > 1.\overline{0}$, the function requires 91 μs. (6600, July 1975)

AUTHOR B.E. Purcell DATE: July 1975

TITLE

Bessel Functions $J_n(x)$, $Y_n(x)$, $I_n(x)$ and $K_n(x)$ of Integer Order

ENTRY

CALL BESGEN(X,N,BIJ,BKY,M,NCALC)

X real input variable, $0.0 \leq x \leq 667.2$

N integer, input variable, the highest order required, $N \geq 2$

BIJ real output array of length (N+1) to store $I_0(X)$, ..., $I_N(X)$ or $J_0(X)$, ..., $J_N(X)$

BKY real output array of length (N+1) to store $K_0(X)$, ..., $K_N(X)$ or $Y_0(X)$, ..., $Y_N(X)$

M integer, input variable, used to select I's and K's or J's and Y's.

M = +1 returns I_0, ..., I_N in BIJ(1), ..., BIJ(N+1) and K_0, ..., K_N in BKY(1), ..., BKY(N+1).

M = -1 returns J_0, ..., J_N in BIJ(1), ..., BIJ(N+1) and Y_0, ..., Y_N in BKY(1), ..., BKY(N+1).

NCALC integer, output parameter (see EXIT).

ROUTINES CALLED

AI0BES, AI1BES, AK0BES, AK1BES, AJYBES from AELIB

EXIT

The subroutine returns with real results in BIJ and BKY. The number of orders calculated is returned in NCALC.

An error message is issued if X < 0, and BIJ and BKY are set to infinity. NCALC is set to 0.

In cases where N >> X and the high order terms cannot be calculated, a message is issued giving the highest order +1 calculated. NCALC is set to the highest order +1, and the remaining elements of BIJ and BKY are set to infinity.

ACCURACY Normally at least 12 significant figures, except at zeros.

SPEED 1-15 ms depending on the argument and order (6600, August 1974).

AUTHORS A. Perreault and J.M. Blair DATE: August 1974

(BESGEN incorporates the subroutine BESLRI published by D.J. Sookne, National Bureau of Standards, Washington, D.C. 20234).

TITLE Bessel Function $J_{\alpha+k-1}(x)$, k=1, 2, ..., n, of Real Order and Real Argument for $\alpha \geq 0$ and $x \geq 0$.

ENTRY CALL JBESS(ALPHA,N,X,Y)

ALPHA real input variable, order of first member of the sequence $\alpha \geq 0$

N integer input variable; number of J Bessel Functions in the sequence, $N \geq 1$

X real input variable, argument of Bessel Functions

Y real output array of length N to return the sequence of J Bessel Functions

ROUTINES CALLED GAMLN, JAIRY from AELIB

STORAGE 2200_8

EXIT The subroutine returns real results in the first N entries of Y, i.e.

$$Y(K) = J_{\alpha+k-1}(x), \quad K=1,2,\ldots,N$$

Error messages are issued for illegal arguments and the elements of Y are set to infinity or zero depending on the type of illegal argument.

ACCURACY Normally at least 12 significant figures, except at zeros.

SPEED .5 - 20 ms depending on the argument and order (6600 August 1976)

AUTHORS D.E. Amos, S.L. Daniel and M.K. Weston, Sandia Laboratories, Albuquerque, N.M. 87115 DATE: August 1974

TITLE

Bessel Function $I_{\alpha+k-1}(x)$, k=1, 2, ..., n, of Real Order and Real Argument for $\alpha \geq 0$ and $x \geq 0$.

ENTRY

CALL IBESS(KODE,ALPHA,N,X,<u>Y</u>)

KODE integer input variable, a flag to indicate the scaling option.

KODE=1 returns $Y(K) = I_{\alpha+k-1}(x)$, K=1,2,...,N.

KODE=2 returns $Y(K) = e^{-x} I_{\alpha+k-1}(x)$, K=1,2,...N.

ALPHA real input variable, the order of the first member of the sequence $\alpha \geq 0$.

N integer input variable, the number of I Bessel Functions in the sequence.

X real input variable, the Bessel Function argument.

Y real output array of length N to return the sequence of I Bessel Functions.

ROUTINES CALLED

GAMLN from AELIB

EXIT

The subroutine returns real results in Y whose first N components contain the sequence:

$$Y(K) = I_{\alpha+k-1}(x), \quad K=1,2,\ldots,N$$

or

$$Y(K) = e^{-x} I_{\alpha+k-1}(x), \; K=1,2,\ldots,N$$

Error messages are issued for illegal arguments and the elements of Y are set to infinity or zero depending on the type of illegal argument.

ACCURACY

Normally at least 12 significant figures.

SPEED

.5-20 ms depending on the argument and order (6600 August 1974).

AUTHORS

D.E. Amos, S.L. Daniel and M.K. Weston, Sandia Laboratories, Albuquerque, N.M. 87115 DATE: August 1974

TITLE Bessel Functions $I_n(z)$ and $J_n(z)$ of Integer Order and Complex Argument.

ENTRY CALL COMBES(Z,N,BESIJ,M,NCALC)

Z complex, input variable, the Bessel function argument:

$|\text{real}(Z)| < 741.68$ for I's

$|\text{imag}(Z)| < 741.68$ for J's

N integer input variable, the highest order required. $N \geq 0$.

BESIJ complex output array of length (N+1) to store $I_0(z),\ldots\ldots,I_n(z)$ or $J_0(z),\ldots,J_n(z)$.

M integer input variable, used to select I's or J's.

M = +1 returns $I_0(z),\ldots.,I_n(z)$ in BESIJ(1),.........,BESIJ(N+1).

M = -1 returns $J_0(z),\ldots..,J_n(z)$ in BESIJ(1),......,BESIJ(N+1).

NCALC integer, output parameter (see EXIT).

STORAGE 1300_8

EXIT The subroutine returns with complex results in BESIJ. The number of orders successfully calculated is returned in NCALC.

An error message is printed if Z or N is illegal and both real and imaginary elements of BESIJ are set to infinity. NCALC is set to 0.

In cases where $N >> |Z|$ and the high order terms cannot be calculated to required accuracy, NCALC is set to the highest order +1 calculated, and the remaining elements of BESIJ are set to (0.,0.).

ACCURACY At least 11 significant figures.

SPEED 2-40 ms depending on the argument and order
(6600 August 1974)

AUTHORS A. Perreault and J.M. Blair DATE August 1974
(adapted from the subroutine BESLCI published by
D.J. Sookne, National Bureau of Standards,
Washington, D.C. 20239)

TITLE Bessel Functions $J_p(z)$ for $0 \leq p \leq 1$ and Complex z.

ENTRY CALL JBESCPX(P,X,Y,$\underline{\text{U}}$,$\underline{\text{V}}$)

P real input variable, the order of the Bessel function

X,Y real input variables specifying the real and imaginary parts of z; i.e. $z = x + iy$.

U,V real output variables, specifying the real and imaginary parts of $J_p(z)$; i.e. $J_p(z) = u + iv$.

EXIT Normal Exit: The subroutine returns to the calling routine with the values of the real and imaginary parts of $J_p(z)$ in U and V respectively.

Error Exit: If P, X, or Y is infinite or indefinite, or if $p < 0$ or $p > 1$, U and V are set to indefinite, the error message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is printed, and control is returned to the calling routine.

ROUTINES CALLED GAMMA from AELIB, LEGVAR from FORTRAN

STORAGE 2000_8

ACCURACY Approximately 11 significant figures.

SPEED for $p = 0.5$, $t \simeq .5$ ms (6600, November 1975)
for $p \neq 0.5$ and $R \leq 20$, $1.5 \text{ ms} < t < 6 \text{ ms}$. (6600, Nov. 1975)

AUTHOR C.J. Johnson DATE April 1969
Revised by B.E. Purcell DATE November 1975

TITLE

The Airy Function and its Derivative

$$Ai(x) = \frac{1}{3}\sqrt{x}[I_{-1/3}(\xi) - I_{1/3}(\xi)]$$

$$Ai(-x) = \frac{1}{3}\sqrt{x}[J_{1/3}(\xi) + J_{-1/3}(\xi)]$$

$$Ai'(x) = -\frac{1}{3}\,x[I_{-2/3}(\xi) - I_{2/3}(\xi)]$$

$$Ai'(-x) = -\frac{1}{3}\,x[J_{-2/3}(\xi) - J_{2/3}(\xi)]$$

$$\xi = \frac{2}{3}\,|x|^{3/2}$$

ENTRY

CALL JAIRY(X,RX,XI,AI,DAI)

X real input variable, the Airy function argument

RX real input variable, RX = $\sqrt{|x|}$

XI real input variable, XI = $\frac{2}{3}\,x^{3/2}$

AI real; output value of the Airy function at x.

DAI real; output value of the derivative of the Airy function at x.

STORAGE 1200_8

EXIT The subroutine returns the real value of the Airy function in AI and the real value of the derivative of the Airy function in DAI. No error messages are issued.

ACCURACY ∿13 significant figures.

SPEED ∿185 ms (6600 March 1975)

AUTHORS D.E. Amos, S.L. Daniel and M.K. Weston
Sandia Laboratories, Albuquerque, N.M.
87115 DATE: March 1975

TITLE Complex Bessel Function $Y_0(z)$.

ENTRY C = CBESY0(Z)

Z - complex input variable, $Z \neq 0$.

C - complex, the Bessel Function value, $Y_0(z)$.

ROUTINES CALLED JBESCPX from AELIB, LEGVAR from FORTRAN.

EXIT Normal Exit: If z is a legal argument, the function returns $Y_0(z)$ to the calling routine.

Error Exit: If z is indefinite, infinite or zero, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is issued and control is returned to the calling routine.

ACCURACY $|Y_0(z)|$ is accurate to at least 10 digits except at zeroes of the function. If the real and imaginary parts of $Y_0(z)$ are of the same order of magnitude, each is accurate to at least 10 digits. If not, the larger is accurate to at least 10 digits and the smaller to the same number of decimal places as the larger.

SPEED If $|z| < 12$, the function requires 2.8 ms (6600, Sept. 1975)

If $|z| \geq 12$, the function requires 2.0 ms (6600, Sept. 1975)

AUTHOR B.E. Purcell DATE Sept. 1975

TITLE Complex Bessel Function $Y_1(z)$.

ENTRY C = CBESY1(Z)

Z - complex input variable, $Z \neq 0$

C - complex, the Bessel Function value $Y_1(z)$.

ROUTINES CALLED JBESCPX from AELIB, LEGVAR from FORTRAN.

EXIT Normal Exit: If z is a legal argument, the function returns $Y_1(z)$ to the calling routine.

Error Exit: If z is indefinite, infinite, or zero, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BESSEL FUNCTION

is issued, and control is returned to the calling routine.

ACCURACY $|Y_1(z)|$ is accurate to at least 10 digits except at zeroes of the function. If the real and imaginary parts of $Y_1(z)$ are of the same order of magnitude, each is accurate to at least 10 digits. If not, the larger is accurate to at least 10 digits and the smaller to the same number of decimal places as the larger.

SPEED If $|z| < 12$, the function requires 3.2 ms (6600, September 1975)

If $|z| \geq 12$, the function requires 2.1 ms (6600, September 1975)

AUTHOR B.E. Purcell DATE September 1975

TITLE

Bickley Function $Ki_1(x) = \int_x^\infty K_0(t)dt$, $x \geq 0$, the integral of the modified Bessel Function $K_0(t)$.

ENTRY

R = AKI1(X)

X - real input variable, $x \geq 0$.

ROUTINES CALLED

LEGVAR from FORTRAN

EXIT

Normal Exit: If x is a legal argument, the function returns $Ki_1(x)$ to the calling routine.

Error Exit: If x is negative, indefinite or infinite, the function value is set to indefinite, the message

***ILLEGAL ARGUMENT FOR BICKLEY FUNCTION

is issued and control is returned to the calling routine.

ACCURACY

At least 13 significant figures.

TIMING

$$\left.\begin{array}{ll} 0 \leq x < 1 & 34\ \mu s \\ x \geq 1 & 37\ \mu s \end{array}\right\} \text{for Cyber 170 Model 175 (March 1977)}$$

AUTHOR

P. Christie & J.M. Blair DATE March 1977

TITLE

Bickley Function $Ki_3(x) = \int_x^\infty Ki_2(t)dt$, $x \geq 0$, the integral of the Bickley Function $Ki_2(t)$. (See write-up for KIN for complete definition of Bickley Functions, $Ki_n(x)$.)

ENTRY

R = AKI3(X)

X - real input variable, $x \geq 0$.

ROUTINES CALLED

LEGVAR from FORTRAN.

EXIT

Normal Exit: If x is a legal argument, the function returns $Ki_3(x)$ to the calling routine.

Error Exit: If x is negative, indefinite or infinite, the function value is set to indefinite, the message:

***ILLEGAL ARGUMENT FOR BICKLEY FUNCTION

is issued and control is returned to the calling routine.

ACCURACY

At least 13 significant figures.

TIMING

$$\left.\begin{array}{ll} 0 \leq x \leq 1 & 36\ \mu s \\ 1 < x & 37\ \mu s \end{array}\right\}$$ for Cyber 170, Model 175 (March 1977)

AUTHOR

P. Christie & J.M. Blair   DATE   March 1977

TITLE

Bickley Functions $Ki_n(x)$, n=1,2,...,10 for $x \geq 0$.
$Ki_n(x)$ is defined as $Ki_n(x) = \int_x^\infty Ki_{n-1}(t)dt$, n=1,2,3,...
where $Ki_0(x) = K_0(x)$.

ENTRY

CALL KIN(X,AKIN)

X real input variable, $x \geq 0$.

AKIN real, output array of length 10 to store $Ki_n(x)$, n=1,2,...,10.

ROUTINES CALLED

LEGVAR from FORTRAN

EXIT

Normal Exit: If x is a legal argument, the subroutine returns the values of the Bickley functions $Ki_n(x)$, n=1,2,...,10 in AKIN.

Error Exit: If x is negative, indefinite or infinite, the elements of the array AKIN are set to indefinite, the message

***ILLEGAL ARGUMENT FOR BICKLEY FUNCTION

is issued and control is returned to the calling routine.

ACCURACY

At least 12 significant figures.

TIMING

$0 \leq x < 1$	90 μs	for Cyber 170, Model 175 (March 1977)
$1 \leq x < 6$	86 μs	
$x \geq 6$	84 μs	

AUTHOR

P. Christie & J.M. Blair DATE March 1977

TITLE Legendre Polynomials $P_n(x) = \frac{1}{2^n} \sum_{m=0}^{n/2} (-1)^m \binom{n}{m} \binom{2n-2m}{n} x^{n-2m}$

ENTRY CALL PN(X,A,M)

X real input variable

A real output array of length(N+1) which stores the polynomials.

M integer input variable, =N+1, where N = highest order polynomial required.

EXIT The polynomials are stored in A with $A(1) = P_0(x),\ldots,$ $A(N+1) = P_n(x)$. No error messages are issued.

ACCURACY 13 significant figures

SPEED ∿.01 N ms (6600, October 1972)

AUTHOR J.H. Schmidt DATE October 1972

TITLE Associated Legendre Function

$P_n^m(\theta)$ for $0 \leq \theta \leq \pi$ and $m=0,1,2,\ldots,n$

ENTRY CALL ALEGEND(THETA,A,L,N)

THETA real input variable, the argument in radians
A real output array of length (N+1) to return the polynomials
N integer input variable, the highest order of polynomial required
L integer input variable, L=N+1

EXIT The polynomials are stored in A with $A(1) = P_n^0$;

$A(2) = P_n^1,\ldots,A(N+1) = P_n^n$

No error messages are issued.

ACCURACY Approximately 12 significant figures. Considerably less near multiples of $\pi/2$.

SPEED For $n \geq 2$, $0 < \theta < \pi$; .15 + .015 N ms (6600, Aug. 1969)

AUTHOR T.T. Tan DATE August 1969

TITLE Chi-square Probability Function

$$Q(x|n) = [2^{n/2}\Gamma(n/2)]^{-1} \int_x^\infty t^{(n/2)-1} e^{-t/2}\,dt$$

ENTRY R=CHISQ(X,N) $X \geq 0$, $N \geq 1$.

X real input variable, the chi-square value whose probability is to be tested.

N integer input variable, the number of degrees of freedom associated with X.

EXIT CHISQ returns a real result; the probability P, $0.0 \leq P \leq 1.0$, of obtaining a value $\geq$ X, given that X has a χ^2 distribution with N degrees of freedom.

The following error messages may be printed and the function set to negative indefinite:

(1) If X < 0.0

'ERROR IN CHISQ - ATTEMPTING TO TEST A NEGATIVE CHI-SQUARED ESTIMATE'

(2) If N < 1

'ERROR IN CHISQ - DEGREES OF FREEDOM LESS THAN 1'

(3) If N > 320 or if X > 200

'ERROR IN CHISQ - PARAMETER(S) TOO LARGE'

ROUTINES CALLED GAMIN and GAMMA from AELIB.

ACCURACY ∿12 or 13 significant figures.

AUTHORS C.H. Kerr and J.M. Blair DATE April 1974

TITLE Student's t-distribution

$$A(x|n) = [\sqrt{n}\ B(1/2,n/2)]^{-1} \int_{-\infty}^{x} (1 + t^2/n)^{-(n+1)/2}\, dt \quad \text{(one-tailed)}$$

$$A(x|n) = [\sqrt{n}\ B(1/2,n/2)]^{-1} \int_{-|x|}^{|x|} (1 + t^2/n)^{-(n+1)/2}\, dt \quad \text{(two-tailed)}$$

ENTRY R=TDISTN(X,N,TAIL)

X real input variable, the t value to be tested.

N integer input variable, the number of degrees of freedom associated with X. $N \geq 1$.

TAIL integer input variable, a flag to indicate whether a one- or two-tailed probability is desired.

TAIL=1 one-tailed probability
TAIL=2 two-tailed probability

EXIT TDISTN returns a real result; probability P, $0 \leq P \leq 1.0$, of obtaining a result .

(a) $\leq X$ if TAIL=1

(b) $\leq |X|$ if TAIL=2

given that X has a student's t-distribution with N degrees of freedom.

If $N < 1$, the message

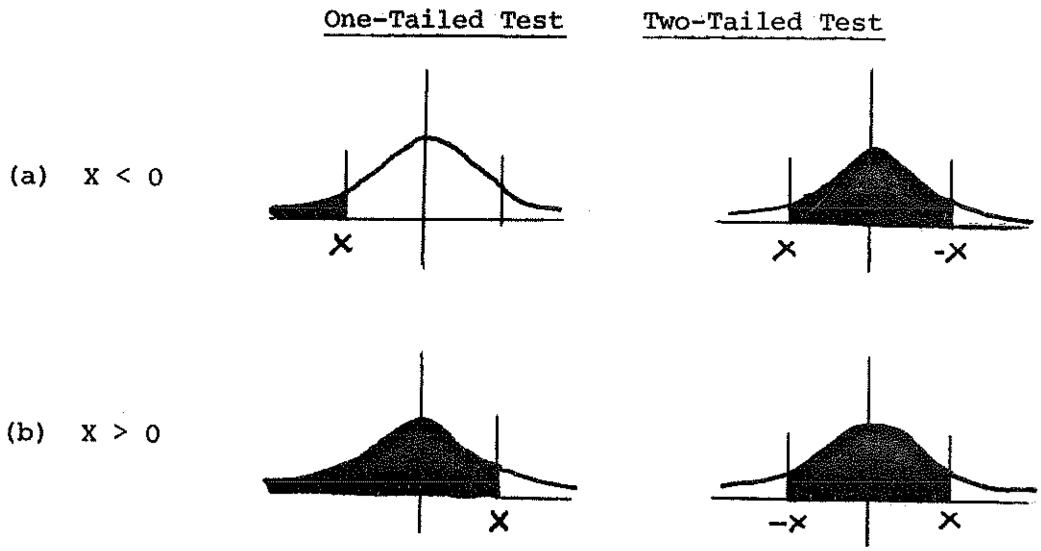
'ERROR IN TDISTN - DEGREES OF FREEDOM LESS THAN 1'

is printed and the value of the function is set to negative indefinite.

ACCURACY ∿12 or 13 digits.

EXAMPLES

In the following diagrams, the shaded area under the curve represents the probability TDISTN that will be calculated for the given value of X.



AUTHORS C.H. Kerr and J.M. Blair DATE April 1974

TITLE Inverse Hyperbolic Sine, arcsinh(x), for All Real x.

ENTRY R=ARSINH(X)

X real input variable.

ROUTINES CALLED PRESET from AELIB.

EXIT The function returns to the calling program, storing in R the real value of arcsinh(x). No error messages are issued.

ACCURACY ∿14 significant figures.

SPEED ∿240 µs (6600, June 1974).

REFERENCE C.W. Clenshaw, Mathematical Tables, Vol. 5, pages 9, 26, 1962.

AUTHORS J.M. Blair and S. Berube DATE June 1974

1- 3 PHYSICS FUNCTIONS

00-09 SPECIAL INTEGRALS

1-3-00	RESIN	RESONANCE INTEGRAL
1-3-01	GLINT	GAUSSIAN/LORENTZIAN CONVOLUTION

10-19 COULOMB WAVE FUNCTIONS

1-3-10	COULOMB	COULOMB WAVE FUNCTIONS

20-29 CLEBSCH-GORDON,RACAH(W) AND X COEFFICIENTS

1-3-20	CLEBSCH	FCLEBSH CLEBSCH-GORDAN COEFFICIENT
1-3-21	WCOEF	FWCOEF RACAH OR W-COEFFICIENT
1-3-22	XCOEF	X-COEFFICIENT

TITLE Resonance Integral

$$\psi(x,t) = \frac{1}{\sqrt{4\pi t}} \int_{-\infty}^{\infty} \frac{e^{-(x-y)^2/4t}}{1+y^2}\,dy \quad \text{for } -\infty < x < \infty,\ 0 \leq t < \infty$$

ENTRY R=RESIN(X,T); X,T are real input variables.

EXIT This function returns the real value of $\psi(x,t)$.

ACCURACY ∿13 significant figures.

SPEED 0.8 ms for $t<(0.5/2\pi)^2$; 1.0 ms for $t\geq(0.5/2\pi)^2$ (6600, September 1969).

AUTHOR C.J. Johnson DATE September 1969

TITLE

Convolution Integral of Gaussian and Lorentzian Functions.

$$G = \int_{-\infty}^{\infty} \frac{e^{-(E'-E)^2/2\sigma^2}}{\varepsilon_1^2 + (E'-E_1)^2} dE'$$

ENTRY

CALL GLINT(E,E1,EPS1,SIGMA,G)

E real, location parameter E of Gaussian.

E1 real, location parameter E_1 of Lorentzian.

EPS1 real, non-zero, width parameter ε_1 of Lorentzian, (half width at half maximum).

SIGMA real, non-zero, width parameter σ of Gaussian, (standard deviation).

G real, value of convolution integral.

ROUTINES CALLED

LEGVAR from FORTRAN.

EXIT

Normal Exit: The subroutine returns to the calling routine with the value of the integral in G.

Error Exit: If E, E1, EPS1 or SIGMA is infinite or indefinite or if EPS1 or SIGMA is zero, G is set indefinite, the error message

```
*** ILLEGAL VARIABLE FOR GLINT
```

is printed and control is returned to the calling routine.

ACCURACY

Relative accuracy is better than 1 in 10^{12}, unless

$|(E-E_1)/\sigma| \geq 17$ or $|\varepsilon_1/\sigma| \geq 28$, approximately, in which

case the relative accuracy is better than 1 in 10^7. If this case occurs, the message

```
*** THIS BRANCH OF GLINT GIVES REL. ACCURACY BETTER THAN
    1 IN 10**7
```

is printed before control is returned to the calling routine.

SPEED Depends on the value of $(E-E_1)/\sigma$, approximately $1.25 + 0.18|(E-E_1)/\sigma|$ milliseconds per evaluation, over a wind range of input variables. (6600, June 1976)

AUTHORS B.E. Purcell and W.N. Selander DATE November 1975

TITLE

Coulomb Wave Functions $F_\ell(\eta,\rho)$, $F_\ell'(\eta,\rho)$, $G_\ell(\eta,\rho)$, $G_\ell'(\eta,\rho)$ for $0 \leq \eta \leq 12$ and $\ell \leq 50$

ENTRY

CALL COULOMB(RHO,ETA,LP,F,FP,G,GP)

RHO real input variable to specify the argument ρ.

ETA real input variable to specify the argument η.

LP integer input variable with value = L+1, specifying that function values are to be calculated for ℓ=0 to L.

F,FP,G,GP real output arrays of length LP to return the wave function values.

EXIT

Tables of $F_\ell(\eta,\rho)$, $F_\ell'(\eta,\rho)$, $G_\ell(\eta,\rho)$, $G_\ell'(\eta,\rho)$ for ℓ=0 to L are stored in the arrays, F, FP, G, GP. If ρ=0 and η or L≠0, the answer spaces are set to zero and "COULOMB FAILS" together with the values of ρ,η, and L is printed. This also happens when η>12.

STORAGE

1500_8

ACCURACY

Approximately five significant figures when ρ is small, otherwise about five decimal places.

SPEED

Variable, depending on the values of ρ, η and ℓ. Typical speeds to obtain the four functions are:

.3 ms for ℓ=0 and 2.0 ms for ℓ>0 (6600, June 1969).

REFERENCES

(1) A. Tubis, "Tables of Nonrelativisitic Coulomb Wave Functions", L.A. 2150 (1958) p.29.

(2) C.E. Froberg, "Numerical Treatment of Coulomb Wave Functions", Revs. Modern Phys. 27, 399 (1955).

AUTHOR C.J. Johnson DATE June 1969

TITLE

Clebsch-Gordan Coefficient, $C(j_1,j_2,j_3;\ m_1,m_2,m_3)$ (uses Condon and Shortley Phase convention).

ENTRY

```
R=CLEBSCH(RJ1,RJ2,RM1,RM2,RJ3)
```

where all arguments are real, or a high-speed version may be called by

```
R=FCLEBSH(N1,N2,N3,N4,N5)
```

where all arguments are integer and

```
N1 = RJ1 + RJ2 - RJ3
N2 = RJ1 - RM1
N3 = RJ2 - RM2
N4 = RJ1 + RM1
N5 = RJ2 + RM2
```

Note: CLEBSCH must be called to set up tables of logarithms and square roots before FCLEBSH can be used.

COMMON BLOCKS USED

CLEBSCH contains three common blocks CGRACAH (162 words), KLEBSYS (9 words) and TRIANG (1 word).

EXIT

This function returns the value of $C(j_1,j_2,j_3;m_1,m_2,m_3)$ to the calling routine.

CLEBSCH checks the argument list for illegal parameters. A zero function value is returned in case of

i) Triangle condition failure,
ii) Array size exceeded, i.e. (RJ1+RJ2+RJ3) > 78,
iii) Other selection rule failures.

An error message is always printed for the latter two errors, but is only printed for a triangle condition failure if the first (and only) word in the labelled common block TRIANG is set to 1 (default value is 0).

FCLEBSH does no testing and does not issue any messages.

SPEED CLEBSCH ∿.22 ms

FCLEBSH ∿.12 ms

(6600, August 1971)

ACCURACY All coefficients were accurate to 12 digits; for some 13 and even 14 digits were obtained (reference 1).

REFERENCES

(1) M. Rotenberg, R. Bivins, et al., The 3-j and 6-j Symbols, Technology Press, MIT, 1959.

(2) L.C. Biedenharn and H. van Dam, Quantum Theory of Angular Momentum, Academic Press, 1965, p. 246 and 296.

AUTHORS J.H. Schmidt and R.Y. Cusson DATE August 1971

TITLE

Racah or W-Coefficient

$$W(abcd,ef) = (-1)^{a+b+c+d} \begin{Bmatrix} a & b & e \\ d & c & f \end{Bmatrix}$$

The curly backets contain Wigner's 6-j symbol.

ENTRY

```
R=WCOEF(A,B,C,D,E,F)
```

where all the arguments are real; or a high-speed version where all the arguments are integers may be called by

```
F=FWCOEF(I1,I2,I3,I4,I5,I6)

I1 = A + B + C + D
I2 = A + D + E + F
I3 = A + B + E
I4 = C + D + E
I5 = A + C + F
I6 = B + D + F
```

Note: WCOEF must be called to set up tables of logarithms before FWCOEF can be used.

COMMON BLOCKS USED

WCOEF uses four COMMON blocks, CGRACAH (162 words), RACSYS (14 words), XRACAH (2 words) and TRIANG (1 word).

EXIT

This function returns the value of W(abcd,ef).

WCOEF checks all selection rules and program array sizes. If illegal parameters are encountered, the value 0. is returned. An error message is printed, except when the triangle condition fails. If this is the case, the message appears only if the first word in the labelled common block TRIANG is set to 1 (default value is 0).

FWCOEF does not check the Regge symbol for negative integers. The mode of exit is unpredictable if this entry is used with illegal parameters, even reasonable looking results may be generated.

ACCURACY

12 significant figures at least (Reference 1).

SPEED

WCOEF ∿.25 + .015 M ms

FWCOEF ∿.20 + .015 M ms

where M = MIN{I3,I4,I5,I6} - MAX{I1,I2,b+c+e+f} (6600, May 1972).

REFERENCES

(1) M. Rotenberg, et al., The 3-j and 6-j Symbols, Technology Press, MIT, 1959.

(2) J. Schwinger, Quantum Theory of Angular Momentum, L.C. Biedenharn and H. van Dam, Editors, Academic Press, 1966, p. 229 and 298.

AUTHORS R.Y. Cusson and J.H. Schmidt DATE May 1972

TITLE X-Coefficient on Wigner 9-j Symbol

$$[X] = X \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & k \end{pmatrix}$$

ENTRY R=XCOEF(A,B,C,D,E,F,G,H,K)

where A,...,K are real.

ROUTINES CALLED

WCOEF from AELIB.

EXIT This function returns the value of the X-coefficient. If illegal parameters are encountered, the value 0. is returned and an error message is printed.

ACCURACY $\sim$12 significant figures.

SPEED $\sim$.75 + (ZMAX-ZMIN) ms

where ZMIN = MAX(|B-F|,|A-K|,|D-H|)
ZMAX = (B+F,A+K,D+H)

(6600, April 1972).

AUTHORS R.Y. Cusson and J.H. Schmidt DATE April 1972

1- 4 ENGINEERING FUNCTIONS

1-4 ENGINEERING FUNCTIONS

This section contains subroutines for the evaluation of various functions which are useful in engineering problems. It is not intended to include methods for the solution of special engineering problems, such as heat transfer or fluid flow.

The initial set of subroutines included in this revision (i.e., Revision D) are for the evaluation of several thermodynamic or transport properties of light water as functions of thermodynamic variables. These "stand-alone" subroutines are intended for rapid, efficient evaluation of properties for the user who requires only single properties and who wants to avoid the overhead incurred in using one of the standard thermodynamic or transport "packages" available at CRNL. A more complete description of routines supported by the Mathematics and Computation Branch is given in the branch document MCTD-25, Thermodynamic and Transport Property Subroutines for Light Water.

W.N. Selander
1983 May 31

TITLE Saturation pressure of light water as a function of temperature.

INTRODUCTION Function PSATLW returns the saturation pressure of light water at a given temperature. The formula for PSATLW is given in [1], page 17. PSATLW is the β_K function multiplied by the critical pressure in megapascals.

ENTRY R=PSATLW(TDEGC)

TDEGC real input variable, temperature in degrees Celsius.

R real output variable into which the saturation pressure in megapascals is stored.

EXIT Normal Exit: If $0.0 \leq \text{TDEGC} \leq 374.15$ then the function returns the corresponding saturation pressure.

Error Exits: If TDEGC is infinite or indefinite, the function value is set to indefinite, the message

```
*** ILLEGAL INPUT VARIABLE IN PSATLW
```

is issued and control is returned to the calling program.

If TDEGC is outside the closed interval [0.0, 374.15] then the function value is set to indefinite, the message

```
*** INPUT TEMPERATURE = x OUT OF RANGE IN PSATLW
```

(where x is the input argument) is issued and control is returned to the calling program.

ACCURACY This function reproduces each value in Table 1 of the ASME Steam Tables [1].

SPEED

27 μsec	(175 May 1982)
64 μsec	(6600 May 1982)
210 μsec	(720 May 1982)

SUPPORT DIFFERENCES An FTN4 version of this routine is not available. PSATLW is supported on AELIB5 on NOS and NOS/BE.

REFERENCE

[1] Meyer C.A., McClintock R.B., Silvestri G.J., Spencer R.C., "ASME Steam Tables: Thermodynamic and Transport Properties of Steam", ASME, New York, 1967.

AUTHOR M.D. Kent Date 1982 May

TITLE Saturation temperature of light water as a function of pressure.

INTRODUCTION Function TSATLW returns the saturation temperature of light water at a given pressure. TSATLW is a rational function approximation to the inverse of function PSATLW (see routine 1-4-00). This means of approximation produces code which is eight times faster than conventional run-time inversion methods.

ENTRY R=TSATLW(PMPASC)

PMPASC real input variable, pressure in megapascals.

R real output variable into which the saturation temperature in degrees Celsius is stored.

EXIT Normal Exit: If $0.610801E-03 \le PMPASC \le 22.12$ then the function returns the corresponding saturation temperature in degrees Celsius.

Error Exits: If PMPASC is infinite or indefinite, the function value is set to indefinite, the message

```
*** ILLEGAL INPUT VARIABLE IN TSATLW
```

is issued and control is returned to the calling program.

If PMPASC is outside the closed interval [0.610801E-03, 22.12] then the function value is set to indefinite, the message

```
*** INPUT PRESSURE = x OUT OF RANGE IN TSATLW
```

(where x is the input argument) is issued and control is returned to the calling program.

ACCURACY When converted to British units, TSATLW agrees with Table 2 of [1] when three decimals are retained. Rounding off to two decimals causes an error of ± 0.01°F at eight points within the table. This anomaly is illustrated in the following example.

Pressure	=	2120.0	PSI
Exact Temperature	=	644.1150	°F
ASME Temperature	=	644.12	°F
TSATLW Temperature	=	644.1148	°F

The TSATLW temperature, when rounded to three decimals, gives 644.115, and when rounded to two decimals, gives 644.11°F, which differs from the ASME reference value by 0.01, TSATLW produces a rounded triple-point temperature of 0.01°C, however, when the unrounded function value is converted to British units a value of 32.017°F is obtained.

SPEED

42 μsec	(175 May 1982)
109 μsec	(6600 May 1982)
240 μsec	(720 May 1982)

SUPPORT DIFFERENCES

An FTN4 version of this routine is not available. TSATLW is supported on AELIB5 on NOS and NOS/BE.

REFERENCE

[1] Meyer C.A., McClintock R.B., Silvestri G.J., Spencer R.C., "ASME Steam Tables: Thermodynamic and Transport Properties of Steam", ASME, New York, 1967.

AUTHOR M.D. Kent Date 1982 May

TITLE Surface tension of light water as a function of temperature.

INTRODUCTION Function STENLW returns the surface tension of light water at a given temperature. The formula for STENLW is given in [1], pg. 173.

ENTRY R=STENLW(TDEGC)

TDEGC real input variable, temperature in degrees Celsius.

R real output variable into which the surface tension in Newton per meter is stored.

EXIT Normal Exit: If $0.01 \leq TDEGC \leq 374.0$ then the function returns the corresponding surface tension.

Error Exits: If TDEGC is infinite or indefinite, the function value is set to indefinite, the message

```
*** ILLEGAL INPUT VARIABLE IN STENLW
```

is issued and control is returned to the calling program.

If TDEGC is outside the closed interval [0.01, 374.0] then the function value is set to indefinite, the message

```
*** INPUT TEMPERATURE = x OUT OF RANGE IN STENLW
```

(where x is the input argument) is issued and control is returned to the calling program.

ACCURACY The accuracy of this function varies with temperature. From 0.01 to 180°C the function is accurate to 0.5%. The function loses accuracy steadily up to a maximum of 3% at 360°C. There is an anomaly at one point, namely 370°C, where the value given in the skeleton table in Ref. [1] is approximately 10% larger than the computed value.

SPEED
37 μsec (175 May 1982)
92 μsec (6600 May 1982)
240 μsec (720 May 1982)

SUPPORT DIFFERENCES An FTN4 version of this routine is not available. STENLW is supported on AELIB5 on NOS and NOS/BE.

REFERENCE [1] E. Schmidt, "Properties of Water and Steam in SI Units", (2nd Edition) Springer Verlag, Berlin, 1979.

AUTHOR M.D. Kent Date: May 1982

TITLE

Thermal conductivity of light water and steam as a function of temperature and volume.

INTRODUCTION

Function TCONLW returns the thermal conductivity of light water and steam at a given temperature and volume. The formula for TCONLW is given in [1], pg. 1235, and in [2] pg. 189.

ENTRY

R=TCONLW(TDEGC,VM3KG)

TDEGC real input variable, temperature in degrees Celsius.

VM3KG real input variable, specific volume in meters cubed per kilogram.

R real output variable into which the thermal conductivity in milli-Watts per Kelvin meter is stored.

EXIT

Normal Exit: If $0.01 \leq \text{TDEGC} \leq 1500.0$ then the function returns the corresponding thermal conductivity. The formula imposes no limitations on the volume provided it is positive.

Error Exits: If TDEGC or VM3KG is infinite or indefinite, or if VM3KG is non-positive, the function value is set to indefinite, the message

*** ILLEGAL INPUT VARIABLE IN TCONLW

is issued and control is returned to the calling program.

If TDEGC is outside the closed interval [0.01, 1500.0] then the function value is set to indefinite, the message

*** INPUT TEMPERATURE = x OUT OF RANGE IN TCONLW

(where x is the input argument) is issued and control is returned to the calling program.

ACCURACY

Approximately 4% or better over the range of temperature and volume, approximately 2% in the water region. See the references for further details. The formula used in this subroutine does not always appear to give values of conductivity within the tolerances given in the skeleton table in Ref. [1].

SUPPORT DIFFERENCES

An FTN4 version of this routine is not available. TCONLW is supported on AELIB5 on NOS and NOS/BE.

SPEED

190 µsec	(175 Dec 1982)
1.44 msec	(720 Dec 1982)

REFERENCE [1] E. Schmidt, "Properties of Water and Steam in SI Units", (2nd Edition), Springer Verlag, Berlin, 1979.

[2] Yata J., Minamiyama T. "An Equation for the Thermal Conductivity of Water and Steam", Bulletin of the JSME, Vol. 22, Paper No. 171-10, pg. 1234, September 1979.

AUTHOR M.D. Kent Date 1982 December

TITLE

Dynamic viscosity of light water and steam as a function of temperature and volume.

INTRODUCTION

Function VISLW returns the dynamic viscosity of light water and steam at a given temperature and volume. The formula for VISLW is given in [1], pg. 188.

ENTRY

R=VISLW(TDEGC,VM3KG)

TDEGC	real input variable, temperature in degrees Celsius.
VM3KG	real input variable, volume in meters cubed per kilogram.
R	real output variable into which the dynamic viscosity in kilograms per second meter is stored.

EXIT

Normal Exit: If $0.01 \le \text{TDEGC} \le 800.0$ then the function returns the corresponding dynamic viscosity. The formula imposes no limitations on the volume provided it is positive.

Error Exits: If TDEGC or VM3KG is infinite or indefinite, or if VM3KG is non-positive, the function value is set to indefinite, the message

```
*** ILLEGAL INPUT VARIABLE IN VISLW
```

is issued and control is returned to the calling program.

If TDEGC is outside the closed interval [0.01, 800.0] then the message

```
*** INPUT TEMPERATURE = x OUT OF RANGE IN VISLW
```

(where x is the input argument) is issued and control is returned to the calling program.

ACCURACY

Approximately 3% or better over the range of temperature and volume. For water below 250°C the accuracy is approximately 1%. See Ref. [1] for further details.

SPEED

54 μsec	(175 May 1982)
0.4 msec	(720 May 1982)

SUPPORT DIFFERENCES

An FTN4 version of this routine is not available. VISLW is supported on AELIB5 on NOS and NOS/BE.

REFERENCE

[1] E. Schmidt, "Properties of Water and Steam in SI Units", (2nd Edition) Springer Verlag, Berlin, 1979.

AUTHOR

M.D. Kent Date 1982 May

1- 5 RANDOM NUMBER GENERATION

00-09 EXPONENTIAL DISTRIBUTION GENERATOR

10-19 NORMAL RANDOM VARIABLE GENERATOR

1-5

1-5 RANDOM NUMBER GENERATION

The random number generators in this chapter are more properly termed pseudo-random, since the sequences generated are strictly determinate and reproducible, being derived by well-defined mathematical operations. The sequences have the appearance of randomness, in the sense that there is no obvious relationship between one number and the next, and they pass certain tests for randomness, such as the chi-square test, in which the observed frequency distribution of the numbers is compared with the theoretical distribution; the run test, in which the distribution of runs up or down of a given length is tested; and the serial correlation test, for the correlation between the numbers and their successors.

The generators are satisfactory for many applications, since the properties tested in the above tests are those generally required in practice. However, the user should realize that only certain aspects of randomness, and only limited quantities of numbers, of the order of 250,000, with specific starting values, have been tested. If the application demands some other requirement, or uses longer sequences of numbers, more extensive testing should be done.

Uniformly Distributed Random Numbers

The FORTRAN library function RANF(n)[1] returns numbers distributed uniformly in the range (0,1) (with the end points excluded). RANF uses a multiplicative congruential generator. The multiplier and seed supplied by Control Data Corporation have been replaced by those recommended in [2]. However, the user may reset the seed by calling the subroutine RANSET[1].

Numbers y_i distributed uniformly in the range (a,b) may be obtained from the x_i in (0,1) by the transformation

$$y_i = a + (b-a)x_i$$

Uniformly distributed integers may be derived by a similar transformation of the x_i. For example, to obtain integers n_i in the range [0,99] inclusive, the transformation is

$$n_i = [100x_i]$$

where [x] denotes the integral part of the real number x. The transformation may be carried out in FORTRAN by the statement

```
          NI = 100.*XI.
```

The IMSL Library [3] contains a (0,1) uniform number generator GGUB. We do not recommend this generator since (a) it uses 32-bit arithmetic, as opposed to the 48-bit arithmetic used by RANF, and hence produces poorer distributions of points in two, three and higher dimensional space, and (b) it is an external subprogram, and hence is slower than RANF, which is an intrinsic function.

Exponentially Distributed Random Numbers

Exponentially distributed random numbers have a probability density function

$$P(x) = \frac{1}{\mu} e^{-x/\mu} , \; x \geq 0$$

where μ is the mean of the distribution. That is, the probability of obtaining numbers in the range [x,x+dx] is P(x)dx.

The library function RANEX generates exponentially distributed numbers with $\mu=1$. It uses a table lookup method with linear interpolation, which is accurate to between 3 and 5 digits.

The library subroutine REX generates N exponentially distributed numbers with $\mu=1$, where N is specified by the user [4]. The exponential distribution is divided into a number of regions and a method of sampling these based on an acceptance rejection technique is used. This subroutine is considerably faster if a sequence of random numbers is required, and more accurate than RANEX, and is recommended for use in all new programs. REX uses RANF and so the sequence of random numbers generated may be changed by resetting the seed using the subroutine RANSET [1].

Exponentially distributed numbers x_i with mean μ may be obtained from the deviates y_i produced by RANEX or REX by the transformation

$$x_i = \mu y_i .$$

The IMSL Library [3] contains several exponential deviate generators. We do not recommend their use unless high accuracy is required.

Normally Distributed Random Numbers

Normally distributed random numbers have a probability density function

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} ,$$

where μ is the mean of the distribution and σ the standard deviation.

The library contains a function, RANORM, to compute unit normal deviates, for which $\mu=0$ and $\sigma=1$. RANORM uses a table lookup method with linear interpolation, which is accurate to between 3 and 5 digits. It uses an in-line function to obtain uniformly distributed numbers.

The library subroutine RNORM generates N unit normal deviates for which $\mu=0$ and $\sigma=1$, where N is specified by the user [4]. The normal distribution is divided into four subregions and a method of sampling these based on an acceptance rejection technique is used. This subroutine is considerably faster if a sequence of random numbers is required and more accurate than RANORM, and is recommended for use in all new programs. RNORM uses RANF and so the sequence of random numbers may be changed by resetting the seed using the subroutine RANSET [1].

Normally distributed numbers x_i with mean μ and standard deviation σ may be obtained from the unit normal deviates y_i produced by RANORM or RNORM by the transformation

$$x_i = \mu + \sigma y_i.$$

The IMSL Library [3] contains random unit normal deviate generators. Unless high accuracy is required, we do not recommend their use, as they are based on the 32-bit (0,1) uniform generator GGUB, and are slower than RANORM.

Other Generators

Reference [3] contains a comprehensive list of routines to generate random deviates obeying other statistical distributions. All of the generators use the IMSL uniform generator GGUB, which is an external function, and consequently they are rather slow. Improvements in speed could be made by replacing the calls to GGUB by calls to RANF.

J.M. Blair
1981 May

Revised by C.A. Wills
1982 March

References

[1] FORTRAN Extended Version 4 Reference Manual, Revision E, Publication No. 60497800, Control Data Corp., 1979.

[2] J.E. Mulvihill and J.M. Blair, *Multiplicative Random Number Generators for 48-Bit Computers*, Atomic Energy of Canada Limited report No. AECL-5819, 1977.

[3] IMSL Library Reference Manual.

[4] K.R. Chaplin and C.A. Wills, *Sampling from the Normal and Exponential Distributions*, Atomic Energy of Canada Limited, Report AECL-7428, 1982.

TITLE Unit Mean Exponential Deviate Generator.

ENTRY R=RANEX(I)

I - odd integer, starting number used to generate the sequence of exponentially distributed random numbers X_i with unit mean ($0 < I < 2^{48} - 1$). Suitable numbers are 798293 and 30169. The next random integer in the range $(0,2^{48}-1)$ is stored at I.

EXIT This function returns the value of the next real random number. The corresponding integer is stored at I, so a sequence of random real numbers may be generated by using the statement R = RANEX(I) in a loop.

SPEED 21 μs (6600, June 1969).

REFERENCE J.P. Nicholls, Random Number Generators, Atomic Energy of Canada Limited report AECL-3476 (1969).

AUTHOR J.P. Nicholls DATE June 1969

TITLE Generator for a sequence of Unit Mean Exponential Deviates (This routine is faster and more accurate than RANEX, 1-5-00, and is recommended for use in all new programs.)

ENTRY CALL REX (X,N)

X(N) Real output array, containing the exponentially distributed random numbers.

N Integer input variable, the number of random numbers to generate.

EXIT Normal Exit: Control is returned to the calling routine with the N random numbers in the array X.

Error Exit : If N is less than or equal to zero the first element of X is set to indefinite, the message

***VALUE OF N NOT GREATER THAN ZERO

is issued and control is returned to the calling routine.

ACCURACY Accurate to machine precision.

SPEED Average time to calculate one random number is 4.17 μs (CYBER 170 Model 175, August 1979).

REFERENCE K.R. Chaplin and C.A. Wills, Sampling from the Normal and Exponential Distributions, Atomic Energy of Canada Limited, Report AECL-7428, 1982.

AUTHOR K.R. Chaplin Date 1979 August

TITLE Unit Normal Deviate Generator.

ENTRY R=RANORM(I)

I - odd integer, starting number used to generate the sequence of normally distributed random numbers X_i with zero mean and unit variance $(0<I<2^{48}-1)$. Suitable numbers are 798293 and 30169.

EXIT This function returns the value of the next random real number. The corresponding integer is stored at I, so a sequence of random numbers may be generated by using the statement R=RANORM(I) in a loop.

SPEED 25 µs (6600, August 1969).

REFERENCE J.P. Nicholls, Random Number Generators, Atomic Energy of Canada Limited report AECL-3476 (1969).

AUTHOR J.P. Nicholls DATE August 1969

TITLE Generator for a sequence of Unit Normal Deviates (This routine is faster and more accurate than RANORM, 1-5-10, and is recommended for use in all new programs.)

ENTRY CALL RNORM (X,N)

X(N) Real output array, containing the normally distributed random numbers

N Integer input variable, the number of random numbers to generate.

EXIT Normal Exit: Control is returned to the calling routine with the N random numbers in the array X.

Error Exit : If N is less than or equal to zero the first element of X is set to indefinite, the message

```
***VALUE OF N NOT GREATER THAN ZERO
```

is issued and control is returned to the calling routine.

ACCURACY Accurate to machine precision.

SPEED Average time to calculate one random number is 3.91 μs (CYBER 170 Model 175, August 1979)

REFERENCE K.R. Chaplin and C.A. Wills, Sampling from the Normal and Exponential Distributions, Atomic Energy of Canada Limited, Report AECL-7428, 1982.

AUTHOR K.R. Chaplin Date 1979 August

1- 6 STATISTICAL TESTING (FOR FUTURE USE)

1-6 STATISTICAL TESTING

IMSL[1] contains a wide range of statistical subroutines which we recommend to users. The relevant chapters of this library are listed here for reference:

Chapter A - Analysis of Experimental Design Data
Chapter B - Basic Statistics
Chapter C - Categorized Data Analysis
Chapter M - Mathematical and Statistical Special Functions
Chapter N - Nonparametric Statistics
Chapter O - Observation Structure
Chapter S - Sampling

L.E. Evans
1981 May

Reference

[1] IMSL Library Reference Manual.

1- 7 MATRIX AND VECTOR MANIPULATION

00-09 BASIC MATRIX AND VECTOR OPERATIONS

10-19 SPARSE MATRIX UTILITIES

1-7 MATRIX AND VECTOR MANIPULATION

This section of AELIB contains some basic matrix manipulation routines and two sparse matrix routines. Chapter V of IMSL[1] contains some more specialized matrix utilities and Chapter U provides some very useful input/ output routines for vectors and matrices.

A matrix is said to be sparse if it contains many zeroes. The utility routine PACKER removes these zeroes and condenses such a matrix into a special packed form which is common to all AELIB sparse matrix routines. This routine may be used to prepare input matrices for the linear system solvers SPARSE/SPARSEB and PDSPARL and for the nonlinear system solver NLSPAR.

The routine JACOB defines a sparse Jacobian matrix for a user's function and is also called internally by NLSPAR.

L.E. Evans
1981 May

Reference

[1] IMSL Library Reference Manual.

TSUM
ASUM
ITSUM
IASUM

TITLE Series Summation Functions Capable of Handling Real and Integer Arrays as well as Positive and Alternating Signs.

ENTRIES (1) Summation of Real Series

R=TSUM(Y(I),N) to compute $\sum_{j=i}^{i+n-1} y(j)$

R=ASUM(Y(I),N) to compute $\sum_{j=i}^{i+n-1} (-1)^{j-i} y(j)$

y a real input array containing values to be summed.

I,N integer input variables supplying values for i and n respectively.

(2) Summation of Integer Series

I=ITSUM(L(I),N) to compute $\sum_{j=i}^{i+n-1} \ell(j)$

I=IASUM(L(I),N) to compute $\sum_{j=i}^{i+n-1} (-1)^{j-i} \ell(j)$

L an integer input array containing values to be summed.

I,N integer input variables supplying values for i and n respectively.

EXIT The function returns a real result for TSUM and ASUM; an integer result is returned for ITSUM and IASUM. No error checking is done.

SPEED Depends on array size. For a ten-element array, about .1 ms (6600, July 1970).

AUTHOR M.B. Carver DATE July 1970

TITLE

Multiply, Add or Subtract Real Matrices.

ENTRY

Multiplication: CALL MMULT(P,Q,R,N,M,L,NP,NQ,NR) computes R=P*Q

Addition: CALL MADD(P,Q,R,NP,NQ,NR,N,M) computes R=P+Q

Subtraction: CALL MSUB(P,Q,R,NP,NQ,NR,N,M) computes R=P-Q

P real, input matrix dimensioned (NP,MP).

Q real, input matrix dimensioned (NQ,MQ).

R real, result matrix, dimensioned (NR,MR).

NP,NQ,NR
integer, number of rows of P, Q and R respectively, as dimensioned in the calling program.

N,M,L
for MMULT: integer, submatrices actually used are P(N,M), Q(M,L), R(N,L).

N,M for MADD and MSUB: integer, actual number of rows (N) and the actual number of columns (M) of P, Q and R to be used.

Note: L=1 gives matrix by vector multiplication.

EXIT

The new matrix generated is stored in R. Portions of R not used remain unchanged.

Note: For addition and subtraction, R may be the same array as P or Q.

SPEED

MADD and MSUB	.083 ms for N=M-4	(6600, July 1969)
MMULT	.588 ms for N=M=L=4	

EXAMPLE

Given matrices of the following sizes:

P(5,5) Q(10,9) R(4,8)

To add the upper left-hand 3 x 4 portions of P and Q:

```
CALL MADD(P,Q,R,5,10,4,3,4)
```

To multiply the portions P(4,2) and Q(2,7):

```
CALL MMULT(P,Q,R,4,2,7,5,10,4)
```

AUTHOR A.E. Russon DATE August 1969

TITLE Multiply, Add or Subtract Complex Matrices

ENTRY

CALL CMMULT(P,Q,R,NP,NQ,NR,K,L,M,N) computes R=P*Q
CALL CMADD(P,Q,R,NP,NQ,NR,K,L,M,N) computes R=P+Q
CALL CMSUB(P,Q,R,NP,NQ,NR,K,L,M,N) computes R=P-Q

P(K,L) complex input matrix

Q(M,N) complex input matrix

R complex output matrix whose dimensions depend on those of P and Q.

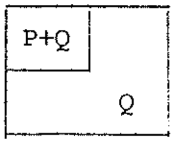
NP integer, input variable. Number of rows of P as dimensioned in the calling program.

NQ integer, input variable. Number of rows of Q as dimensioned in the calling program.

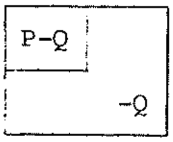
NR integer, input variable. Number of rows of R as dimensioned in the calling program.

Matrices of different sizes may be added or subtracted as long as the larger matrix completely contains the smaller. For example, if P is completely contained in Q, R is computed as follows:

CMADD:R=P+Q=



CMSUB:R=P-Q=



ROUTINES CALLED SETBAD, ALERPR, TRACEB and LIBSTAT from AELIB.

EXIT A normal exit returns control to the calling program with results of the computation stored in matrix R. Matrices P and Q are unchanged. If an error is detected in the arguments passed, a diagnostic message is printed with traceback and control returns to the calling routine with the entries of matrix R set to indefinite. The possible errors are:

```
***INPUT PARAMETER VIOLATION.  ALL INTEGER INPUT VARI-
   ABLES MUST BE > 0.
   NP=    NQ=    NR=    K=    L=    M=    N=

***ARRAY BOUNDS EXCEEDED.  K > NP OR M > NQ IS INVALID.
   K=    NP=    M=    NQ=

***ARRAY BOUNDS EXCEEDED.
   FOR CMMULT K > NR IS INVALID.
   FOR CMADD/CMSUB MAX(K,M) > NR IS INVALID.
   NR=        K OR MAX(K,M)=

***MATRIX MULTIPLICATION ERROR.  COLUMNS IN P.NE.ROWS IN Q.
   NUMBER OF COLUMNS IN P=   NUMBER OF ROWS IN Q=

***MATRIX SIZES IN ERROR.
   SMALLER MATRIX NOT COMPLETELY CONTAINED IN LARGER.
   P(  BY  )   Q(  BY  )
```

TIMING

Running the example below on MFA using 10^5 calls to each CMMULT, CMADD and CMSUB yields the following approximate times per call to each routine:

```
APPROXIMATE TIME/CALL TO CMMULT        464.20 µs
APPROXIMATE TIME/CALL TO CMADD         177.74 µs
APPROXIMATE TIME/CALL TO CMSUB         170.45 µs
```

EXAMPLE

```
      PROGRAM TCMULT(INPUT,OUTPUT)
C
C  THIS EXAMPLE ILLUSTRATES HOW TO SET UP A CALL TO CMMULT, CMADD AND
C  CMSUB.  SUBROUTINE ASSIGN IS A USER DEFINED SUBROUTINE WHICH
C  INITIALIZES THE ENTRIES IN COMPLEX MATRICES X(A,B) AND Y(C,D).
C  NX, NY AND NZ ARE THE NUMBER OF ROWS IN COMPLEX MATRICES X, Y AND Z
C  RESPECTIVELY .
C
      COMPLEX X(10,10),Y(10,10),Z(10,10)
      INTEGER NX,NY,NZ,A,B,C,D
      NX = NY = NZ = 10
      A = 3
      B = C = 5
      D = 7
      CALL ASSIGN(X,Y,NX,NY,A,B,C,D)
      CALL CMMULT(X,Y,Z,NX,NY,NZ,A,B,C,D)
      CALL CMADD(X,Y,Z,NX,NY,NZ,A,B,C,D)
      CALL CMSUB(X,Y,Z,NX,NY,NZ,A,B,C,D)
      STOP
      END
```

CMMULT
CMADD
CMSUB

AUTHOR	P.H. Green Revised by G.L. Klawitter	DATE	September 1970 February 1980

TITLE Determinant of a Real Matrix by Pivotal Condensation with Partial Pivoting.

ENTRY R=DET(A,N,NRA)

A real N*N input matrix

NRA integer, number of rows in A as dimensioned in the calling program.

EXIT This function returns the real value of the determinant. The matrix A is destroyed.

SPEED $\simeq 3 * N^2$ μs (6600, October 1969).

AUTHOR C.J. Johnson DATE October 1969

TITLE Complex Determinant by Pivotal Condensation with Partial Pivoting.

ENTRY C=CDET(A,N,NRA)

A complex N*N input matrix.

NRA integer, number of rows in A as dimensioned in the calling program.

EXIT This function calculates the complex value of the determinant and returns it as the value of the function. The matrix A is overwritten.

If CDET is called with N<1 or N>NRA, CDET is set to zero, an error message printed and control returned to the calling program.

ACCURACY Depends on degree of ill-conditioning of matrix A and order N.

SPEED 0.01 N^3 ms for large N (6600, 1970 August).

AUTHOR P.H. Green DATE August 1970

TITLE

Store a Sparse Matrix in Packed Form Suitable for Manipulation by Sparse Matrix Routines.

ENTRY

CALL PACKER(A,B,IRN,IP,M,N,IA)

A real output array, dimension IA to receive packed matrix.

B real input array, dimension(M,N) containing matrix to be packed.

IRN integer output array, dimension IA to receive row cross references.

IP integer output array, dimension N+1 to receive column cross references.

M integer input variable, the number of rows in B.

N integer input variable, the number of columns in B.

IA integer input variable, the dimension of A and IRN should be > number of non-zeros.

ADDITIONAL ENTRY INFORMATION

The output unit LP (default 6) or the round off error EPS (default 10^{-14}) may be changed by including the common block /SPARS/LP,EPS,IS,JS. A non-zero is defined if $|B(I,J)| \geq$ EPS. IS and JS are required by other sparse routines.

COMMON BLOCKS USED

SPARS

EXIT

A, IRN and IP are filled according to the requirements of AELIB subroutine SPARSE.

IP(N+1) returns the index of the first unused location in A and IRN and is, therefore, the number of non-zeros plus one.

Should IA be too small, a message is sent to unit LP, and IP(1) is set to -1.

AUTHOR M.B. Carver DATE October 1974

TITLE JACOB: Evaluate and Store a Sparse Jacobian Matrix in Packed Form.

INTRODUCTION Given the subroutine FUNC which calculates the functions

$f_i(x_1,x_2,...,x_n,t)$, i=1,m

JACOB evaluates an approximation to the Jacobian matrix A where

$$a_{ij} = \frac{\partial f_i}{\partial x_j}$$

and stores it in sparse form compatible with other sparse matrix routines.

ENTRY CALL JACOB(M,N,IRN,IP,FUNC,X,F,T,H,HMAX,W,A,IA,IG,Z,MBD)

M integer, input variable, number of functions f_i.

N integer input variable, number of variables x_j.

IRN(IA) integer output array dimension IA, to hold row indices of the packed matrix, A.

IP(NIP) integer output array, dimension NIP $\geq$ N+1, to hold column indices of the packed matrix, A.

FUNC the routine which calculates f_i, assumed to be of the form

SUBROUTINE FUNC(M,N,X,F,T),

which should be written to evaluate

$f_i(x_1,...,x_n,t)$ for i=1,M

The statement "EXTERNAL FUNC" must appear in the calling program.

X(N) Real input array of dimension N, for the value $x = (x_1,...,x_n)$ at which Jacobian is to be calculated.

F(M) real input array of dimension M, for function values $f_i(x_1,\ldots,x_n,t)$ i=1,...,m at which Jacobian is to be calculated.

T real input variable, the value of independent variable, t, at which Jacobian is to be calculated.

H(N) real array, dimension N for the initial perturbations to be used in the calculation of $\partial f/\partial x$. JACOB may alter values in this array.

HMAX(N) real input array, dimension N for the upper limit permitted on each H. If HMAX(1) is set <0, then all steps H are limited to |HMAX(1)| and the dimension of HMAX may be one. If on entry MBD<0, HMAX is not used.

W(IW) real array, dimension IW ≥ M+N, used for workspace.

A(IA) real output array, dimension IA, to receive the packed Jacobian. If on entry MBD<0, A is not used.

IA integer input variable, the dimension of A and IRN in the calling program. If the Jacobian has NZ non-zero elements, IA must be ≥ NZ+1.

IG(ING) integer output array, dimension ING ≥ 2N+1 used to index the most efficient way to evaluate A on the second and subsequent calls to JACOB. If on entry MBD<0, IG is not used.

Z(N) real array, dimension N used for work space.

MBD integer input variable, used for operation control, set to

<0 for constructing the sparsity pattern of the Jacobian for use with other sparse routines.

=0 for initial call, see below.

>0 for subsequent call.

ADDITIONAL ENTRY INFORMATION

On entry, dimensions of all arrays must be declared and values for M, N, X, F, T, IA, MBD, all H, and if MBD>0, at least HMAX(1) should be defined. FUNC must be supplied. On initial call with MBD$\leq$0, the routine evaluates the sparsity structure and sets up arrays IRN and IP. In addition when MBD=0, the array IG is set up and must not be invalidated for subsequent calls; also an initial approximation to the derivatives is calculated and stored in A. On all subsequent calls with MBD>0, the routine evaluates and packs A with a minimum of function calls.

On initial entry with MBD$\leq$0, the variable X(1) is changed to X(1) + H(1) and the corresponding changes to f are taken to indicate the non-zeros in the first column of the Jacobian matrix. X(1) is then restored to its original value and the same procedure is applied to each column. Note that it is important to avoid any special values (e.g. zeros) of X(J), J=1,2,...,N, which cause freak zeros in the Jacobian matrix and it is important that the steps H(J) be large so that all appropriate changes to f are noticeable.

Unless otherwise requested, the routine estimates A by doing a one-sided perturbation of H_j to each X_j. For greater accuracy, two-sided adjustment with optimized selection of the H_j ($\leq$HMAX) is performed.

To obtain this, include the sparse matrix common block.

COMMON/SPARS/LP,EPS,ISCALE,IPRINT

LP is an integer denoting the unit number of auxiliary printout in other sparse matrix routines. The default is 6.

EPS precision to be assumed, i.e. a non-zero element is defined if $|A_{ij}|\geq$EPS. Default is 10^{-14}.

ISCALE is default zero, and must be set >0 to obtain two-sided adjustment described above. (ISCALE<0 adjusts matrix values as required by routine STIFFZ and should not be used elsewhere.)

IPRINT is not required by JACOB but must be included if the common block is used.

EXIT

Normal Exit: The routine exits with the pointer arrays IRN and IP used to index the Jacobian matrix. When MBD>0, the routine also exits with the packed matrix in A and the additional pointer array IG. W may be used for other purposes between calls of JACOB.

Error Exit: If IA is too small to accommodate all non-zero elements, the message

```
***ERROR RETURN FROM JACOB BECAUSE IA IS TOO SMALL
```

is printed, IP(1) is set to -1 and control is returned to the calling routine.

OTHER ROUTINES CALLED

FUNC provided by user.

COMMON BLOCKS USED

SPARS

STORAGE 1100_8

EXAMPLE

The following example program calls JACOB for the functions

$$f(i) = x(i)^2 - x(i-1)$$

and gives Jacobian elements alternating 2x and -1.

```
      PROGRAM JACK(OUTPUT)
C
C     TEST PROGRAM TO ILLUSTRATE USE OF JACOB
C
      REAL  X(10),F(10),W(20),Z(10),A(20),H(10)
      INTEGER IRN(20),IP(11),IG(21)
C
C     SET INITIAL VALUES OF X AND H, HMAX NEG TO SET ALL H
C     LIMITS SAME
C     THEN CALL FUNC TO SET UP INITIAL VALUES OF F
C
      IA=20
      M=N=10
      MBD=0
      HMAX=-1.
      DO 10 I=1,N
      H(I)=.001
```

```
   10 X(I)=I
      CALL FUNC(M,N,X,F,T)
C
C     NOW CALL JACOB WITH MBD=0 TO GET STRUCTURE AND
C     JACOBIAN ELEMENTS
C
      CALL JACOB(M,N,IRN,IP,FUNC,X,F,T,H,HMAX,W,A,IA,IG,Z,MBD)
C
C     PRINT RESULTING MATRIX
C
      PRINT 100,A
  100 FORMAT(10G12.3)
      END

      SUBROUTINE FUNC(M,N,X,F,T)
      REAL X(N),F(M)
C
C     JACOBIAN MATRIX WILL BE ALTERNATING I AND -1 IF X
C     ARE PRESET TO I
C     INDEPENDENT VARIABLE T IS NOT USED IN THIS CASE
C
      DO 10 I=2,M
   10 F(I)=X(I)**2-X(I-1)
      F(1)=X(1)**2
      END
```

REFERENCE

Curtis, A.R., Powell, M.J.D., and Reid, J.K., "On the Estimation of Sparse Jacobian Matrices", AERE Report TP476, 1972.

AUTHOR	Adapted from the Harwell Routine TD02A by M.B. Carver; revised by D.E. Smith	DATE	March 1975 Revised July 1975

1- 8 LINEAR EQUATIONS AND MATRIX INVERSION

00-09	GENERAL MATRICES		
	1-8-00	SOLVEQN	SOLVE A REAL LINEAR SYSTEM (IN CORE)
	1-8-01	CSOLVEQ	SOLVE A COMPLEX LINEAR SYSTEM
	1-8-04	OCSOLVE	SOLVE A REAL LINEAR SYSTEM (OUT OF CORE)
10-19	BANDED MATRICES		
	1-8-10	TRIEQN	SOLVE A REAL TRIDIAGONAL SYSTEM
20-29	LARGE SPARSE MATRICES		
	1-8-20	SPARSE	ANALYZE A SPARSE MATRIX
	1-8-21	SPARSEB	OPERATE ON SPARSE A WITH VECTOR V
	1-8-23	PDSPARL	SPARSE SYMMETRIC POS DEF SYSTEM SOLVER

1-8 LINEAR EQUATIONS AND MATRIX INVERSION

This section deals mainly with the solution of simultaneous linear equations of the form

$$AX = B$$

where A is an n x n coefficient matrix, B is an n x r matrix of r right-hand sides, and X is an n x r matrix of the r solution vectors, and with the calculation of A^{-1}, the inverse of A. No accuracy checks are performed on the computed solutions. The routines in this section solve general real, general complex, banded and sparse linear systems. Note that SPARSE and SPARSEB are to be used together as a sparse linear system solver.

If x_c is the computed solution of the equations $Ax = b$, where b is a column vector, the magnitude of the residual vector $r = Ax_c-b$ is not a measure of the accuracy of x_c. To determine the accuracy of x_c we need to know the accuracy of A and b and the condition number of A, or we can perturb A and b and recalculate x.

Chapter L of [1] contains a comprehensive list of subroutines for the solution of linear equations and matrix inversion, including a number of routines which perform accuracy checks, for full, symmetric, and banded matrices.

J.M. Blair
1981 May

Reference

[1] IMSL Library Reference Manual.

TITLE

Real Matrix Inversion and Linear Equations Solution by Gaussian Elimination with Partial Pivoting.

ENTRY

CALL SOLVEQN(A,$\underset{\sim}{B}$,N,NR,M,NRA,NRB)

A real N x N input array containing, for M=0, the matrix to be inverted, and, for M=1, the matrix of coefficients of the system of linear equations.

B real N x N or N x NR array used for input and output containing, for M=0, the required matrix inverse on output, and, for M=1, the NR right-hand sides of the linear equations on input and the NR sets of solutions on output.

N integer, input variable specifying for M=0 the order of the square matrix A, and, for M=1, the number of equations.

NR integer input variable, for M=0, set NR equal to N; for M=1, NR ($\geq$1) is the number of right-hand sides of the linear equations and the number of sets of solutions.

M integer input variable; M=0 selects matrix inversion, M=1 selects equation solution.

NRA integer input variable, the number of rows in A as dimensioned in the calling program.

NRB integer input variable, the number of rows in B as dimensioned in the calling program.

EXIT

For linear equations, the solutions replace the right-hand sides in B. For matrix inversion, A^{-1} is stored in B.

In both cases, A is destroyed. If the matrix is singular, "SOLVE EQUATIONS FAILS" is printed.

ACCURACY

Depends on the conditioning of the equations.

SOLVEQN

SPEED Approximately 2.9 N^2 (N+3NR) μs for large N (6600, January 1971).

AUTHOR	C.J. Johnson and A.E. Russon	DATE	January 1969
		Revised	January 1971
	Revised by J.M. Blair	DATE	March 1980

TITLE

Linear Equations and Inverse of a Complex Matrix by Gaussian Elimination with Partial Pivoting.

ENTRY

CALL CSOLVEQ(A,B,N,NR,M,NRA,NRB)

A complex N x N input matrix containing the coefficients of the equations.

B complex N x NR input matrix containing the right-hand sides of the equations. For solution of equations set NR to the number of right-hand sides. For inversion, set NR to N. B is also used as an output matrix to return the N x NR array of solutions or the inverse of A.

M integer input control variable;

Use M≠0 to solve a set of equations Ax=B.
Use M=0 to find the inverse of A.

NRA integer, number of rows of A as dimensioned in the calling program.

NRB integer, number of rows of B as dimensioned in the calling program.

EXIT

For solving a set of complex linear equations, the solution(s) replaces the right-hand side(s) in B.

For matrix inversion, A^{-1} is stored in B.

In both cases, A and B are destroyed. If the matrix is singular, "NO SOLUTION" is printed and control returned to the calling program.

If the array dimensioning parameters are invalid, an error message is printed and control is returned to the calling program.

SPEED

Matrix Inversion	0.04 N^3 ms for large N
Solve Equations	0.01 N^3 ms for large N

ACCURACY Depends on the conditioning of the equations and on N.

AUTHOR P.H. Green and J.M. Blair DATE August 1970

TITLE Solution of linear equations Ax = B by Gaussian elimination with partial pivotting, where the matrices A and B are stored out of core.

ENTRY CALL OCSOLVE(X,N,WKA,IER)

X(NRX) real output array in which the solution vector is passed back to the calling routine NRX $\geq$ N.

N Integer input variable, the number of equations.

WKA(NRW) real or integer array, which need only be dimensioned in the calling routine, for use as working storage by OCSOLVE, NRW $\geq$ 3N.

IER integer output variable, error flag passed back by OCSOLVE, with values defined in Error Exits below.

ADDITIONAL ENTRY INFORMATION

The input matrices are assumed to be on mass storage, and intermediate results are stored on ECS. Before calling OCSOLVE, the user must:

(a) Set up the data on TAPE17 using the FTN mass storage input/output routines. In particular, this requires:

i) declaring the files named TAPE17 and TAPE18 on the program statement. TAPE18 is local to the subroutine and is returned when no longer needed.

ii) calling OPENMS to set INDEX.

iii) calling WRITMS to write the columns of A and B on TAPE17 using a numbered index.

(b) Declare sufficient ECS on the job card, and set up the following common blocks for ECS transfers:

```
COMMON/MATRIX1/ECS1(N1)
COMMON/MATRIX2/ECS2(N2)
COMMON/MATRIX3/ECS3(N3)
COMMON/MATRIX4/ECS4(N4)
LEVEL 3, ECS1, ECS2, ECS3, ECS4
```

where $N1 + N2 + N3 + N4 \geq N*(N-1)/2$.

ROUTINES CALLED

MOVLUV from FORTRAN.

COMMON BLOCKS USED

MATRIX1, MATRIX2, MATRIX3, MATRIX4.

STORAGE 2200_8

EXITS

Normal Exit

The subroutine returns to the calling routine with the solution in X, and IER=0. The original matrices A and B on mass storage are left unchanged.

Error Exits

If any element in A or B is indefinite or infinite, X is set to indefinite, IER is set to 129, the message

***ILLEGAL ARGUMENT IN OCSOLVE, INFINITE OR INDEFINITE IN MATRIX OR RIGHT-HAND SIDE

is printed with traceback via a call to AELERR and control is returned to the calling routine.

If A is singular, X is set to indefinite, IER is set to 130, the message

***SINGULAR MATRIX IN OCSOLVE

is issued with traceback via a call to AELERR and control is returned to the calling routine.

ACCURACY

Depends on the conditioning of A. The solution is accurate to approximately 14-K digits, where K is the condition number of A.

TIMING

CP time is approximately $1.82*10^{-6}*N^3$ for $N \geq 300$. IO time is approximately .65*N for $N \geq 300$. For four values of N, the CP and IO times for sample runs were:

N	CP(s)	IO(s)	(6600, May 1976)
288	45	27	
360	86	37	
432	146	44	
576	340	67	

PROGRAM EXAMPLE

This sample program reads the 500*500 matrix and right-hand side column by column from cards, calls the subroutine OCSOLVE and then prints the solution.

Note: MATRIX1 has (500)*(500-1)/2 = 124,750 elements. If more than 131,072 elements had been required, then MATRIX2 and possibly MATRIX3 and MATRIX4 would have to have been dimensioned greater than 1 to store the rest.

```
      PROGRAM TRIAL(INPUT,OUTPUT,TAPE17,TAPE18)
C
C     RESERVE ENOUGH SPACE FOR ECS
C
      COMMON/MATRIX1/A1(124750)
      COMMON/MATRIX2/A2(1)
      COMMON/MATRIX3/A3(1)
      COMMON/MATRIX4/A4(1)
      LEVEL 3,A1,A2,A3,A4
C
      REAL A(500),X(500),WKS(1500),INDEX(502)
C
C     SET UP MASS STORAGE
C
      CALL OPENMS(17,INDEX,502,0)
C
C     READ COEFFICIENT ARRAYS FROM CARDS AND COPY THEM
C     TO MASS STORAGE
C
      DO 10 I=1,500
         READ 100,(A(J),J=1,500)
         CALL WRITMS(17,A,500,I)
 10   CONTINUE
      READ 100,(X(J),J=1,500)
      CALL WRITMS(17,X,500,501)
C
C     SOLVE THE LINEAR SYSTEM USING OCSOLVE AND PRINT
```

```
C     SOLUTION
C
      CALL OCSOLVE(X,500,WKS,IER)
      PRINT 200,(X(J),J=1,500)
  100 FORMAT(F20.10)
  200 FORMAT(1H1,50(10F10.5,1))
      END
```

AUTHOR K.R. Chaplin DATE May 1976

TITLE Tridiagonal Equation Solver by LU Decomposition without Pivoting.

ENTRY CALL TRIEQN(DIAG,SUP,SUB,B,N)

DIAG real one-dimensional input array containing the main diagonal DIAG(1) to DIAG(N).

SUP real one-dimensional input array containing the super diagonal SUP(1) to SUP(N-1).

SUB real one-dimensional input array containing the subdiagonal SUB(1) to SUB(N-1).

B real one-dimensional array to supply the RHS of the system of equations on entry and to return the solution on exit.

N integer input variable, the order of the system.

EXIT TRIEQN exits with the solution vector stored in B. DIAG and SUP are destroyed. No error messages are issued.

SPEED $\sim 4 + .18\ N\ \mu s$ (6600, April 1972)

ACCURACY 12 significant figures. This method is stable with respect to rounding errors if the matrix is diagonally dominant, i.e. if

$$|DIAG(I)| \geq |SUP(I)| + |SUB(I-1)|$$

The accuracy is unpredictable if this condition is not satisfied.

AUTHOR L. Yamazaki DATE April 1972

TITLE Analyze a Sparse Matrix and Perform LU Decomposition Prior to Operating on the Matrix (for Use with SPARSEB).

ENTRY CALL SPARSE(A,IND,IW,N,NP,G,U,IA)

A real input and output array, dimension IA which on input contains the packed matrix stored by columns, with elements in the natural row order of each column. Thus a typical order might be

$$a_{11}, a_{31}, a_{51}, a_{22}, a_{42}, a_{62}, a_{33}, a_{53}, \ldots$$

IND integer input and output array, dimension (IA,2) to specify row cross references. If a_{ij} is held in A(K), then IND(K,1) must contain i. Thus for the above, IND would contain 1, 3, 5, 2, 4, 6, 3, 5. The remaining elements are set internally.

IW integer input and output array, dimension (NP,13) to specify column cross references. IW(J,1) contains the position in A of the first element of each column J, thus IW would contain 1, 4, 7, ... above.

Also IW(N+1,1) should contain the position of the first vacant element in A (i.e. the number of non-zeros plus one). The remaining elements are set internally.

N integer input variable, the number of columns in A.

NP integer input variable, the number of rows dimensioned for IW, $NP \geq N + 1$.

G real output parameter to indicate the success or failure of SPARSE. If SPARSE fails, G is set to -1; otherwise G is set to 0.

U real input variable, a fraction set 0. < U < 1. to control the choice of pivots. During the elimination when searching for a pivot, any element less than U times the largest element in the current row or column is excluded. Thus decreasing U, biases the algorithm towards maintaining sparsity at the expense of accuracy and vice versa. A value of 0.25 is recommended.

IA integer input variable, the dimension of array A. It should be set about 10% higher than the number of non zeros, as the routine may create extra non-zero terms to improve the accuracy of the elimination.

ADDITIONAL ENTRY INFORMATION

The authors claim that better results are obtained when matrix elements are scaled prior to elimination. The elements will be scaled by the routine MSCALE if the user sets JSCALE in the common block:

```
COMMON/SPARS/LP,EPS,ISCALE,JSCALE
```

LP unit to receive messages, default 6.

EPS round off accuracy, default 10^{-14}.

ISCALE used by JACOB.

JSCALE 0 default, no scaling.

>0 scaling will be done by MSCALE.

<0 scaling factors determined from a previous call will be used.

MSCALE is loaded with SPARSE and is activated by setting JSCALE>0. No other action is required.

EXIT

On exit, the arrays A, IND and IW have been rearranged in decomposed form, suitable for use by SPARSEB. IW(N+1) is set to the current number of non zeros plus one. A, IND and IW(1) to IW(N+1,5) should be left undisturbed between calls to SPARSE and SPARSEB but the remainder of IW is required only within SPARSE and may be overwritten on exit.

The following error conditions cause G to be set -1, and messages to be sent to unit 6:

(1) An element in A is out of order due to IW or IND being incorrectly sequenced.

(2) The matrix is singular.

(3) IA is too small.

ROUTINES CALLED

SORTAG from AELIB and MSCALE, a utility routine loaded with SPARSE.

COMMON BLOCKS USED

SPARS

STORAGE 1700_8

REFERENCE Curtis, A.R. and Reid, J.K., "FORTRAN Subroutines for the Solution of Sparse Sets of Linear Equations", AERE R6844, 1971.

AUTHOR Adapted from Harwell Routines MA18A and MC12A by M.B. Carver DATE March 1975

TITLE Operate on a Sparse Matrix A with Vector V (for use with SPARSE).

ENTRY CALL SPARSEB(A,IRN,IP,N,NP,G,MTYPE,VECT,W,NAME)

A real input array dimension IA, previously processed by SPARSE.

IRN integer input array, dimension (IA,2) containing row cross references from SPARSE.

IP integer input array, dimension (NP,5) containing column cross references from SPARSE.

N integer input variable, the number of columns of matrix A.

NP integer input variable, the first dimension of IP, NP $\geq$ N + 1.

G real estimate of error from SPARSE.

MTYPE operation control.

If 1, perform $V = [A^{-1}][V]$

2, perform $V = [(A^{T})^{-1}][V]$

3, perform $V = [A][V]$

4, perform $V = [A^{T}][V]$

6, enter a new matrix A with the same sparsity structure as the previous A by calling subroutine NAME.

5, analyze and decompose the new matrix A.

VECT real array, dimension N to be operated on and to return the result.

W real array, dimension N for internal workspace.

NAME subroutine supplied by the user, used to read in the replacement matrix when MTYPE=5. It should be declared as

SUBROUTINE NAME(N,W,J)

and should return all the non-zero elements A_{IJ} of column J of the new matrix in W(I) without altering the other elements of W. It will be called N times with the J values not necessarily in sequence.

The calling program must contain the statement EXTERNAL NAME.

It is assumed that a given matrix A will be operated on several times by SPARSEB for each call of SPARSE. When operations on a new matrix with the same sparsity structure as A are required, it may be entered via SPARSEB with MTYPE=5 and analyzed via SPARSEB with MTYPE=6. This is faster than performing the entire triangulation process as in SPARSE. For a new matrix of different structure, a call to SPARSE is required to establish a new pivotal sequence. If JSCALE is positive, the new matrix is scaled by MSCALE.

EXIT

The results of a successful operation are stored in VECT. For errors, G is set -1 and a message sent to unit 6. The following errors are reported:

(1) MTYPE out of range.
(2) No operation performed as a previous call to SPARSE had an error.
(3) No operation performed as no previous call to SPARSE was made.
(4) A zero pivot encountered in the new matrix.

ROUTINES CALLED

MSCALE, a utility routine loaded with SPARSEB and NAME, a routine supplied by the user.

COMMON BLOCKS USED

SPARS

STORAGE	1200_8
REFERENCE	Curtis, A.C. and Reid, J.K., "FORTRAN Subroutines for the Solution of Sparse Sets of Linear Equations, AERE R6844, 1971.
AUTHOR	Adapted from the Harwell Routines MA18B, MA18C, MA18D by M.B. Carver
DATE	March 1975

TITLE

Solve a Symmetric Positive-Definite Sparse System of Linear Equations

$$\sum_{j=1}^{N} a_{ij}x_j = b_i, \ i=1,2,...,N$$

(i.e. perform $x = A^{-1}B$ or related problems).

INTRODUCTION

There are three types of entry:

(1) IENT=1, which must always be the first call, decomposes A into LDL^T, where L is lower triangular and D is diagonal, using a pivotal strategy designed to maintain sparsity.

(2) IENT=2 uses the factors produced by (1) (or (3)) to find $A^{-1}B$ or AB.

(3) IENT=3 decomposes a new matrix A_1 of the same pattern as A, using the pivotal sequence determined by an earlier call with IENT=1.

It is envisaged that (3) may be called many times for one call of (1), as it is much faster. Also it is expected that (2) may be called with many different vectors B for the same matrix A, after it has been decomposed by (1) or (3) entry.

ENTRY

CALL PDSPARL(A,IND,IW,N,NP,IA,B,MTYPE,IENT)

A(IA) Real array holding the non-zero elements of the lower-triangular part of the matrix A on entry to (1) or (3), and the elements of D and L on exit. The elements are stored by columns.

For entry to (1), the elements of A must be in natural row order within each column and the columns must be in natural order; that is a_{ij} precedes $a_{k\ell}$ if $j<\ell$ or if $j=\ell$ and $i<k$. Thus a typical order might be a_{11}, a_{31}, a_{51}, a_{22}, a_{42}, a_{33}, a_{53}, a_{44}, a_{55}.

For entry to (3), the elements of a new matrix A_1 with the same sparsity pattern as A have to be loaded in a different way to conform with the lower triangular pattern of the reordered matrix A with pivots in sequence and additional elements created during the elimination (i.e. by a prior call to (1)). The Ith column in this order was column IW(I,2) in the original matrix; the Ith column of the new matrix starts at A(IW(IW(I,2),1)) and ends at A(IW(IW(I+1,2),1)-1). The element stored in A(k) is in the IND(k)th row of the original matrix. All elements A(k), k=1,2,...,kA where Ka = IW(N+1)-1 must be set (i.e. the Ith column must be varied from 1 to N to set the A parameter for a call to entry (3)).

IND(NRIND)

Integer input array whose first IA elements are used to hold row numbers and whose remaining elements are used for workspace. If a_{ij} is held in A(k) then IND(k) must contain i; for the above example IND would contain 1, 3, 5, 2, 4, 3, 5, 4, 5. IND is altered by a call to entry (1). NRIND $\geq$ 2*IA.

IW(NP,6) Integer input array. Before a call to entry (1), the values of IW(J,1), J=1,2,...,N should be set to the subscript in array A of the Jth diagonal element, and IW(N+1,1) should be set to the subscript of the first unused location in A; thus in the above example, IW would contain 1, 4, 6, 8, 9, 10. ((IW(I,J),I=1,N+1), J=1,2) should be left undisturbed between a (1) entry and a subsequent entry to (2) or (3). The rest of IP is workspace. NP $\geq$ N + 1.

N Integer input variable set to the order of the matrix A.

NP Integer input variable set to indicate the number of rows dimensioned for the array IW which should be $\geq$ N + 1.

IA Integer input variable set to indicate the size of the arrays A and IND, as dimensioned in the calling routine.

For entry to both (1) and (3), IA must be large enough to hold the non-zero elements of the lower triangular part of the matrix plus any elements created during the elimination.

B(NRB) Real array used to hold B on entry and $A^{-1}B$ or AB on exit. Only used when a call to (2) is made. NRB $\geq$ N.

MTYPE Integer input variable controlling the action of entry (2). It should have the value 1 if $A^{-1}B$ is required or 2 if AB is required.

IENT Integer input variable used to control entry to PDSPARL.
IENT=1 decomposes A into LDL^T.
IENT=2 computes $A^{-1}B$ or AB.
IENT=3 factorizes a new matrix A.

EXIT

Normal Exit: Control is returned to the calling program with the factors L and D (such that $LDL^T = A$) stored in the array A and their associated pointer arrays IND and IW if IENT=1 or 3, or with the resultant vector x ($=A^{-1}B$ or AB) stored in array B if IENT=2. Note, between a call with IENT=1 and subsequent calls with IENT=2 or 3 the contents of A, IND, IW, N, NP, and IA should not be altered.

If PDSPARL is called with IENT<1 or IENT>3, the message

***WARNING: IENT≠1/2/3; EXECUTION PROCEEDS AS IF IENT=1

is issued, IENT is set to 1 and execution continues.

Error Exit: Control is returned to the calling program if one of the following five error conditions occur. IW(N+1,2) is set equal to -(error number) to allow the user a decision on subsequent action.

(1) ***THE ELEMENT HELD IN A(...) IS OUT OF ORDER

This message is caused by a sequencing error in the data for a call to PDSPARL with IENT=1. A return is made immediately after the error is found.

(2) ***THE ...TH DIAGONAL ELEMENT IS NOT PRESENT

This message is caused by the sparsity pattern presented to PDSPARL with IENT=1 not containing entries for all the diagonal entries. A return is made immediately after the error is found.

(3) ***IA IS TOO SMALL, SPACE RAN OUT WHEN ELIMINATING ON PIVOT ...

This message appears when there is insufficient room to store a new non-zero element generated in elimination operations using the ith pivot. Thus if i<<n, much more space will probably be needed, but if i is nearly equal to n just a little more may suffice. A return is made immediately after the error is found.

(4) ***ZERO PIVOT FOUND IN ROW ...

This message is caused by PDSPARL with IENT=1 or 3 finding a zero pivot. The matrix presented will not be decomposed but a further matrix of the same sparsity pattern may be presented to PDSPARL with IENT=3 as if a zero pivot had not been found.

(5) ***RESULT MAY BE UNRELIABLE SINCE THERE IS A NEG PIVOT IN ROW ...

This message is caused by PDSPARL (IENT=1/2) finding a negative pivot. Execution is continued but the results may be inaccurate. If no action is taken by the calling program, this message may be repeated by an IENT=3 entry.

(6) ***PREVIOUS ENTRY GAVE ERROR RETURN

There is also an error return, if by inspecting IW(N+1,2), PDSPARL with IENT=2 or 3 finds that an earlier entry gave a failure that will cause this entry to fail. IW(N+1,2) is not altered in this case.

ROUTINES CALLED

SORTAG from AELIB.

COMMON BLOCKS USED

AELERCM

STORAGE 1600_8

SPEED

Depends on the size (N) and the sparsity of the matrix. The time example program USE takes to execute is 0.016 seconds (6600, July 1975).

EXAMPLE

The following program solves the equation Ax=B for

$$A = \begin{pmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix} \text{ and } B = \begin{pmatrix} -1 \\ 3 \\ 3 \\ -1 \end{pmatrix}$$

and for another matrix of similar sparsity pattern

$$A^* = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 3 \end{pmatrix} \text{ and } B^* = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

The lower triangular part of matrix A is stored in the AMATRIX array so as to utilize the AECL routine PACKER to pack A and set up the pointer arrays IND and IW. The lower triangular part of the matrix A* is stored by columns in packed form in the ASTAR array to facilitate the loading of the matrix in the form required by PDSPARL when IENT=3.

```
PROGRAM USE(OUTPUT)
INTEGER IND(18),IW(6,6)
REAL AMATRIX(4,4),A(9),B(4)
REAL ASTAR(7),BSTAR(4)
DATA AMATRIX/4.0,-1.0,3*0.0,4.0,-1.0,3*0.0,4.0,-1.0,3*0,4.0/
DATA B/-1.0,3.0,3.0,-1.0/
DATA ASTAR/3.0,1.0,3.0,1.0,3.0,1.0,3.0/
DATA BSTAR/1.0,-1.0,-1.0,1.0/

C
C         INITIALIZATION
C
     N=4
     NP=6
     IA=9
     MTYPE=1
     CALL PACKER(A,AMATRIX,IND,IW,N,N,IA)
C
C         PERFORM LU DECOMPOSITION
C
     CALL PDSPARL(A,IND,IW,N,NP,IA,B,MTYPE,1)
```

```
C
C         PERFORM B=A^-1B
C
      CALL PDSPARL(A,IND,IW,N,NP,IA,B,MTYPE,2)
C
      PRINT 1000,≠SOLUTION≠,(B(I),I=1,N)
C
C         SPECIFY THE NEW MATRIX -ASTAR- OF THE SAME
C         SPARSITY PATTERN AS THE ORIGINAL MATRIX -A-
C         FOR ENTRY TO PDSPARL WITH IENT=3.
C
      K=1
      DO 10 I=1,N
           INDX1=IW(IW(I,2),1)
           INDX2=IW(IW(I+1,2),1)-1
      DO 5 J=INDX1,INDX2
      A(J)=ASTAR(K)
      K=K+1
5     CONTINUE
10    CONTINUE
C
C         DECOMPOSE NEW MATRIX
C
      CALL PDSPARL(A,IND,IW,N,NP,IA,BSTAR,MTYPE,3)
C
C         PERFORM B* = A*^-1B*
C
      CALL PDSPARL(A,IND,IW,N,NP,IA,BSTAR,MTYPE,2)
C
      PRINT 1000,≠SOLUTION2≠,(BSTAR(I),I=1,N)
1000  FORMAT STATEMENT
      END
```

REFERENCES

[1] J.K. Reid, "Two FORTRAN Subroutines for Direct Solution of Linear Equations Whose Matrix is Sparse, Symmetric and Positive-Definite", AERE-R7119.

AUTHOR Adapted from the Harwell Routine MA17A/B/C by D.E. Smith

DATE July 1975

1- 9 MATRIX EIGENSYSTEM ANALYSIS

1-9 MATRIX EIGENSYSTEM ANALYSIS

Introductory Theory

The subroutines in this chapter compute the eigenvalues, and, optionally, the eigenvectors of a given matrix. The eigenvalues and eigenvectors together are termed the eigensystem.

If A is a square matrix, λ is an eigenvalue and $\underset{\sim}{x}$ is the corresponding eigenvector if they satisfy the equation

$$A\,\underset{\sim}{x} = \lambda\,\underset{\sim}{x} \qquad (1)$$

A consequence of (1) is that if $\underset{\sim}{x}$ is an eigenvector, so also is $c\,\underset{\sim}{x}$, where c is any constant. That is, if we multiply an eigenvector of A by any constant, the resulting column vector is also an eigenvector of A.

An equivalent definition of an eigenvalue λ is that it is a root of the equation

$$\det(A-\lambda I) = 0 \qquad (2)$$

where I is the unit matrix and det denotes the determinant. If A is an nxn matrix, equation (2) can be expressed as a polynomial equation of degree n in λ, from which it follows that an nxn matrix has exactly n eigenvalues and eigenvectors. Since the roots of a polynomial equation are in general complex, even if the coefficients are real, the eigenvalues and eigenvectors of a real or complex matrix are in general complex.

Let $\lambda_1,\lambda_2,\ldots,\lambda_n$ denote the eigenvalues and $\underset{\sim}{x}_1,\underset{\sim}{x}_2,\ldots,\underset{\sim}{x}_n$ the eigenvectors of A. Then

$$A\,\underset{\sim}{x}_i = \lambda_i\underset{\sim}{x}_i, \quad i = 1,2,\ldots,n \qquad (3)$$

The n equations in (3) can be written in the form

$$AX = X\Lambda \qquad (4)$$

where X and Λ are square matrices of the form $X = [\underset{\sim}{x}_1\ \underset{\sim}{x}_2\cdots\underset{\sim}{x}_n]$, $\Lambda = \text{diag}\ (\lambda_1,\lambda_2,\ldots,\lambda_n)$; diag denotes the diagonal matrix with diagonal entries $\lambda_1,\lambda_2,\ldots,\lambda_n$ and zeroes elsewhere.

1-9 MATRIX EIGENSYSTEM ANALYSIS

For certain special matrices, namely real symmetric and complex hermitian, the eigenvalues are known to be real. A real matrix A is symmetric if $A^T=A$, where A^T is the transpose of A. That is, if $a_{jk}=a_{kj}$ for all j,k, where a_{jk} is the j-kth element of A. A complex matrix A is hermitian if $\bar{A}^T=A$, where $\bar{A}$ is the complex conjugate of A. That is, if $\bar{a}_{kj}=a_{jk}$ for all j,k, where $a_{jk}=x+iy$, $\bar{a}_{jk}=x-iy$.

The generalized eigenvalue problem is to find the eigenvalues and eigenvectors of the equation

$$A\,\underset{\sim}{x} = \lambda B\underset{\sim}{x} \qquad (5)$$

where A and B are square matrices. Equation (5) can be reduced to equation (1) by premultiplying by A^{-1} or B^{-1}, but this is not recommended in practice, as it is more accurate to solve (5) directly.

Subroutine Capabilities

Section 1-9 contains subroutines EIGRGEN and EIGCGEN to determine the complete eigensystem for general real and complex matrices, EIGRSYM for real symmetric matrices, and EIGCHER for complex hermitian matrices. Clearly EIGCGEN could be used for all four types of matrices, but it is more efficient to use one of the others if the matrix has the appropriate properties.

The above four subroutines are interfaces to EISPACK[1], a collection of subroutines to perform different stages in the calculation of eigenvalues and eigenvectors. A knowledgable user may wish to call the EISPACK subroutines directly, since he may thereby obtain a more efficient and more accurate code for his particular application. This requires some background in numerical analysis and matrix theory. The EISPACK subroutines are stored on a magnetic tape, and information on how to access particular modules is available from the AELIB librarian.

Chapter E of the IMSL Library[2] contains a comprehensive list of subroutines to solve the eigenvalue problem. In addition to subroutines similar to the four mentioned above, it contains subroutines for the generalized eigenvalue problem, and for performing certain intermediate steps in the calculation of eigenvalues and eigenvectors.

J.M. Blair
1981 May

References

[1] B.T. Smith, J.M. Boyle, B.S. Garbow, Y. Ikebe, V.C. Klema, C.B. Moler, Matrix Eigensystem Routines - EISPACK Guide, Springer-Verlag, 1974.

[2] IMSL Library Reference Manual.

TITLE Calculation of all eigenvalues and optionally, eigenvectors of a real symmetric matrix.

ENTRY CALL EIGRSYM(N,IORDER,AR,NVECS,WORKSTG,NUM,EIGVAL,EIGVEC)

N Integer input variable; row dimension of arrays AR and EIGVEC, as declared in the calling program dimension statement.

IORDER Integer input variable; order of the leading subarray of AR for which the eigensystem is desired. IORDER must be no greater than N.

AR Real input array, an IORDER x IORDER symmetric matrix whose eigensystem is to be computed. Only the main diagonal and lower triangle need be defined. The contents of AR are destroyed by the routine.

NVECS Integer input variable, a flag to indicate whether or not eigenvectors are also desired.

= 0 if only eigenvalues are to be found.
≠ 0 if eigenvectors corresponding to all eigenvalues should also be calculated.

WORKSTG Real; one-dimensional array of size N needed for working storage.

NUM Integer output variable indicating the number of eigenvalues (and eigenvectors) that were calculated.

If 0 < NUM < IORDER, the (NUM+1)th eigenvalue was not obtained after 30 iterations. Those eigenvalues that were calculated will be found in ascending order in the first NUM positions of EIGVAL. Corresponding eigenvectors (if required) will be found in the first NUM columns of EIGVEC.

If NUM=IORDER, the eigenvalues of AR will be found in ascending order in EIGVAL.

EIGVAL Real output one-dimensional array of at least IORDER elements, containing the eigenvalues of AR.

EIGVEC Real output IORDER x IORDER array containing the orthonormal set of eigenvectors of AR, stored by columns; i.e. the jth column of EIGVEC contains an eigenvector corresponding to the jth eigenvalue EIGVAL(J).

If only eigenvalues are desired (NVECS=0), this parameter can be omitted from the calling statement.

ROUTINES CALLED IMTQL1,IMTQL2,SORT,TRED1,TRED2 and TRIDIAG, eigenanalysis utility routines, loaded from AELIB and PRESET from AELIB.

STORAGE 2400_8

EXIT NUM returns the number of eigenvalues that were each calculated in less than 30 iterations. These are returned in ascending order in the first NUM positions of EIGVAL. Eigenvectors, if requested (NVECS≠0) are in the corresponding columns of the array EIGVEC.

If 0 < NUM < IORDER,

one of the following warning messages appears on the output file:

"ONLY NUM EIGENVALUES WERE FOUND"

or, if NVECS≠0,

"ONLY NUM EIGENVALUES AND VECTORS WERE FOUND"

ACCURACY 13 significant figures

SPEED The following results represent average times, in seconds, of five trials, using as input matrices of random numbers from a uniform distribution on (-1,1).

	IORDER				
CALCULATION	20	40	60	80	100
ALL EIGENVALUES	.0626	.2788	.7306	1.5306	2.7366
ALL EIGENVALUES AND EIGENVECTORS	.1606	1.0616	3.334	7.5406	14.2956

FTN OPT=2, 6600, March 1975

Note that, in general, execution time is dependent on the nature of the input matrix and its eigenvalues.

PROGRAM EXAMPLE

The program computes the eigenvalues and eigenvectors of a real, symmetric 4 x 4 matrix. The matrix is listed as the first part of the program output. The eigenvalues and eigenvectors are real.

```
      PROGRAM REALSYM(OUTPUT)
C
C TEST PROGRAM TO TEST EIGRSYM.
C AR IS A 4 X 4 SYMMETRIC MATRIX WITH EIGENVALUES 1.Ø, 2.Ø, 5.Ø, 1Ø.Ø
C
      DIMENSION AR(4, 4), WORKSTG(4), EIGVAL(4), EIGVEC(4, 4)
      DATA ((AR(I, J), J = 1, 4), I = 1, 4)
     ,              / 5.Ø, 4.Ø, 1.Ø, 1.Ø,
     ,                4.Ø, 5.Ø, 1.Ø, 1.Ø,
     ,                1.Ø, 1.Ø, 4.Ø, 2.Ø,
     ,                1.Ø, 1.Ø, 2.Ø, 4.Ø /
      DATA N /4/, IORDER /4/, NVECS /1/
C
      PRINT 9ØØØ, ((AR(I, J), J = 1, IORDER), I = 1, IORDER)
9ØØØ  FORMAT(1H1, 1ØX, "DATA MATRIX", //, (1X, 4F8.1))
C
      CALL EIGRSYM( N, IORDER, AR, NVECS, WORKSTG, NUM, EIGVAL, EIGVEC )
C
      IF ( NUM .NE. IORDER ) STOP "FAILURE IN EIGRSYM"
      PRINT 9Ø1Ø, (EIGVAL(I), I = 1, IORDER)
9Ø1Ø  FORMAT(1X, //, 1X, "EIGENVALUES", //, 1X, 4F13.6, /)
      PRINT 9Ø2Ø, ((EIGVEC(I, J), J = 1, IORDER), I = 1, IORDER)
9Ø2Ø  FORMAT(1X, "EIGENVECTORS", //, (1X, 4F13.6))
      END
```

```
                 DATA MATRIX

           5.0       4.0       1.0       1.0
           4.0       5.0       1.0       1.0
           1.0       1.0       4.0       2.0
           1.0       1.0       2.0       4.0

      EIGENVALUES

           1.000000        2.000000        5.000000        10.000000
      EIGENVECTORS

            .707107         .000000         .316228        -.632456
           -.707107         .000000         .316228        -.632456
           0.000000        -.707107        -.632456        -.316228
           0.000000         .707107        -.632456        -.316228
```

AUTHORS C.H. Kerr and J.M. Blair DATE March 1975

TITLE Calculation of all eigenvalues and optionally, eigenvectors of a real general (i.e. non-symmetric) matrix.

ENTRY CALL EIGRGEN(N,IORDER,AR,NVECS,WORKSTG,NUM,EIGVALR, EIGVALI,EIGVECR,EIGVECI)

N Integer input variable, the row dimension of arrays AR, EIGVECR, and EIGVECI, as declared in the calling program dimension statement.

IORDER Integer input variable, the order of leading sub-array of AR, for which the eigensystem is desired. IORDER must be no greater than N.

AR Real input array, an IORDER x IORDER array containing the real general matrix whose eigensystem is to be computed. The contents of AR are destroyed by the routine.

NVECS Integer input variable, a flag to indicate whether or not eigenvectors are also desired.

= 0 if only eigenvalues are to be found.
≠ 0 if eigenvectors corresponding to all eigenvalues should also be calculated.

WORKSTG Real; one-dimensional array of size 2*N needed for working storage.

NUM Integer output variable indicating the number of eigenvalues (and eigenvectors) that were calculated.

If 0 < NUM < IORDER, the (NUM+1)th eigenvalue was not obtained after 30 iterations. Those eigenvalues that were calculated will be in the 1st NUM positions of (EIGVALR,EIGVALI). Similarly, if requested, the 1st NUM eigenvectors will be found in the 1st NUM columns of (EIGVECR,EIGVECI).

If NUM < 0, NVECS ≠ 0, same as above for eigenvalues. AR is not special tridiagonal, so no eigenvectors could be found.

If NUM=IORDER, all eigenvalues of AR will be found in (EIGVALR,EIGVALI).

EIGVALR
EIGVALI
Real output one-dimensional arrays of at least IORDER elements, containing the real and imaginary parts respectively of the eigenvalues of AR in ascending order of modulus.

EIGVECR
EIGVECI
Real output IORDER x IORDER arrays containing unnormalized eigenvectors of AR stored by columms; i.e. the jth columns of (EIGVECR, EIGVECI) contain the eigenvector corresponding to the jth eigenvalue of (EIGVALR,EIGVALI).

If only eigenvalues are desired (NVECS=0), these two parameters can be omitted from the calling statement.

Note that the eigenvectors are unnormalized, and each may be multiplied by an arbitrary complex number.

ROUTINES CALLED

BALANC, BALBAK, ELMHES, ELTRAN, FIGI, FIGI2, HQR, HQR2, IMTQL1, IMTQL2, SORT and TRIDIAG, eigenanalysis utility routines, loaded from AELIB and PRESET from AELIB.

STORAGE 6000_8

EXIT

NUM returns the number of eigenvalues that were each calculated in less than 30 iterations. The eigenvalues of the real general input matrix AR are complex in general; their real and imaginary parts are returned in the first NUM positions of EIGVALR and EIGVALI, respectively, in ascending order of modulus; i.e.

$$\sqrt{\mathrm{EIGVALR(I)}^2 + \mathrm{EIGVALI(I)}^2}$$

The eigenvectors corresponding to the eigenvalues of AR are also complex. If requested, (NVECS≠0) their real and imaginary parts are returned in the first NUM columns of EIGVECR and EIGVECI, respectively.

(1) If: NUM < 0: The warning message

"ONLY NUM EIGENVALUES AND NO EIGENVECTORS WERE FOUND" appears on the output file. The (-NUM + 1)st eigenvalue required >30 iterations to converge to a solution. Although requested (NVECS≠0) no eigenvectors could be calculated, because AR is not a special tridiagonal matrix.

A special tridiagonal matrix is one of the form

$$\begin{bmatrix} a_1 & c_1 & 0 & 0 & 0 & 0 & 0 \\ b_1 & a_2 & c_2 & 0 & 0 & 0 & 0 \\ 0 & b_2 & a_3 & - & 0 & 0 & 0 \\ 0 & 0 & - & - & - & 0 & 0 \\ 0 & 0 & 0 & - & - & - & c_{n-1} \\ 0 & 0 & 0 & 0 & 0 & b_{n-1} & a_n \end{bmatrix}$$

where $b_i c_i \geq 0$, $i = 1,2,\ldots,n-1$

and $b_i c_i = 0$ implies $b_i = 0$ <u>and</u> $c_i = 0$

(2) 0 < NUM < IORDER: One of the following warning messages appears on the output file:

"ONLY <u>NUM</u> EIGENVALUES WERE FOUND"

or, if NVECS ≠ 0,

"ONLY <u>NUM</u> EIGENVALUES AND VECTORS WERE FOUND"

ACCURACY 13 significant figures

SPEED The following results represent average times, in seconds, of five trials, using as input matrices of random numbers from a uniform distribution on (-1,1)

	IORDER				
CALCULATION	20	40	60	80	100
ALL EIGENVALUES	.2672	1.5592	4.3884	9.2313	18.5876
ALL EIGENVALUES AND EIGENVECTORS	.5208	3.4064	10.2658	22.5178	45.5908

FTN OPT=2, 6600, March 1975

Note that, in general, execution time is dependent on the nature of the input matrix and its eigenvalues.

PROGRAM EXAMPLE

The program computes the eigenvalues and eigenvectors of a real 4 x 4 matrix. The matrix is listed as the first part of the program output. Two of the eigenvalues are real, and two are complex conjugates. The eigenvectors corresponding to the real eigenvalues are real, and the eigenvectors corresponding to the complex eigenvalues are complex conjugates.

```
      PROGRAM REALGEN(OUTPUT)
C
C TEST PROGRAM TO TEST EIGRGEN.
C AR IS A 4 X 4 REAL MATRIX WITH EIGENVALUES 0.03, -1.97 + I, -1.97 - I,
C 3.03
C
      DIMENSION AR(4, 4), WORKSTG(8), EIGVALR(4), EIGVALI(4),
     ,          EIGVECR(4, 4), EIGVECI(4, 4)
      DATA ((AR(I, J), J = 1, 4), I = 1, 4)
     ,               /  0.00,  0.07,  0.27, -0.33,
     ,                  1.31, -0.36,  1.21,  0.41,
     ,                  1.06,  2.86,  1.49, -1.34,
     ,                 -2.64, -1.84, -0.24, -2.01 /
      DATA N /4/, IORDER /4/, NVECS /1/
C
      PRINT 9000, ((AR(I, J), J = 1, IORDER), I = 1, IORDER)
9000  FORMAT(1H1, 10X, "DATA MATRIX", //, (1X, 4F8.2))
C
      CALL EIGRGEN( N, IORDER, AR, NVECS, WORKSTG, NUM, EIGVALR,
     ,                             EIGVALI, EIGVECR, EIGVECI )
C
      IF ( NUM .NE. IORDER ) STOP "FAILURE IN EIGRGEN"
      PRINT 9010, (EIGVALR(I), EIGVALI(I), I = 1, IORDER)
9010  FORMAT(1X, //, 1X, "EIGENVALUES", //, 1X, 4(F7.4, 1X, F7.4, 3X),/)
      PRINT 9020, ((EIGVECR(I, J), EIGVECI(I, J), J = 1, IORDER), I = 1,
     ,                                                       IORDER)
9020  FORMAT(1X, "EIGENVECTORS", //, (1X, 4(F7.4, 1X, F7.4, 3X)))
      END
```

```
          DATA MATRIX

     0.00     .07     .27    -.33
     1.31    -.36    1.21     .41
     1.06    2.86    1.49   -1.34
    -2.64   -1.84    -.24   -2.01

 EIGENVALUES

   .0300  0.0000   -1.9700  1.0000   -1.9700 -1.0000    3.0300  0.0000

 EIGENVECTORS

   .4602  0.0000     .1650   .0081     .1650  -.0081    -.2107  0.0000
  -.1534  0.0000   -.2897  -.8413    -.2897   .8413    -.6322  0.0000
  -.4090  0.0000     .4547   .8493     .4547  -.8493   -1.6858  0.0000
  -.4090  0.0000    1.3202   .0646    1.3202  -.0646     .4214  0.0000
```

AUTHORS C.H. Kerr and J.M. Blair DATE March 1975

TITLE Calculation of all eigenvalues and optionally, eigenvectors of a complex Hermitian matrix.

ENTRY CALL EIGCHER(N,IORDER,AR,AI,NVECS,WORKSTG,NUM,EIGVAL,
EIGVECR,EIGVECI)

N Integer input variable, the row dimension of arrays AR, AI, EIGVECR, EIGVECI, as declared in the calling program dimension statement.

IORDER Integer input variable, the order of leading submatrix of input matrix for which the eigensystem is desired. IORDER must be no greater than N.

AR,AI Real input IORDER x IORDER arrays containing the real and imaginary parts, respectively, of the complex Hermitian matrix whose eigensystem is desired. Only the main diagonal and lower triangle need be defined. Contents of AR and AI are destroyed by the routine.

NVECS Integer input variable, a flag to indicate whether eigenvectors are also desired.

= 0 if only eigenvalues are to be found.
≠ 0 if eigenvectors corresponding to all eigenvalues are also to be calculated.

WORKSTG Real; one-dimensional array of size 3*N needed for working storage.

NUM Integer output variable returning the number of eigenvalues (and eigenvectors) that were calculated.

If 0 < NUM < IORDER, the (NUM+1)th eigenvalue was not obtained after 30 iterations. Those eigenvalues that were calculated, will be in ascending order in the 1st NUM positions of EIGVAL.

Note: They are not necessarily the smallest NUM eigenvalues.

Similarly, if requested, the NUM eigenvectors will be found in the 1st NUM columns of EIGVECR, EIGVECI.

If NUM=IORDER, eigenvalues of (AR,AI) will be found in ascending order in EIGVAL.

EIGVAL — Real output one-dimensional array of at least IORDER elements, containing the eigenvalues of the complex Hermitian matrix, (AR,AI).

EIGVECR
EIGVECI — Real output IORDER x IORDER arrays containing the real and imaginary parts, respectively, of the orthonormal eigenvectors of the system; i.e. the jth columns of (EIGVECR,EIGVECI) contain the eigenvector corresponding to the jth eigenvalue EIGVAL(J).

Note that the eigenvectors are normalized to have unit length; i.e.

$$\sum_{I=1}^{IORDER} [EIGVECR(I,J)**2 + EIGVECI(I,J)**2] = 1,$$

J=1,2,...,NUM

The resulting eigenvectors are defined only up to an arbitrary complex factor $e^{i\theta}$. That is, each eigenvector may be multiplied by a complex number $e^{i\theta_j}$, for any θ_j, and the resulting system of eigenvectors will be orthonormal.

If only eigenvalues are desired (NVECS=0) these two parameters can be omitted from the calling statement.

ROUTINES CALLED — HTRIBK,HTRIDI,IMTQL1,IMTQL2,SORT, eigenanalysis utility routines loaded from AELIB and PRESET from AELIB.

STORAGE — 2400_8

EXIT

NUM returns the number of eigenvalues that were each calculated in less than 30 iterations. The eigenvalues of the complex Hermitian input matrix are real, and are returned in ascending order in the first NUM positions of EIGVAL. The corresponding eigenvectors are complex in general. If requested (NVECS≠0) their real and imaginary parts are returned in the first NUM columns of EIGVECR and EIGVECI, respectively.

NORMAL EXIT: NUM=IORDER

ERROR EXIT: 0 < NUM < IORDER

One of the following warning messages appears on the output file:

"ONLY NUM EIGENVALUES WERE FOUND"

or, if NVECS=Ø

"ONLY NUM EIGENVALUES AND VECTORS WERE FOUND"

ACCURACY

13 significant figures

SPEED

The following results represent average times, in seconds, of five trials, using as input matrices of random numbers from a uniform distribution on (-1,1).

	IORDER				
CALCULATION	20	40	60	80	100
ALL EIGENVALUES	.08	.4826	1.3978	3.0394	5.6188
ALL EIGENVALUES AND EIGENVECTORS	.2416	1.561	4.8704	10.9936	21.1502

FTN OPT=2, 6600, March 1975

Note that, in general, execution time is dependent on the nature of the input matrix and its eigenvalues.

PROGRAM EXAMPLE

The program computes the eigenvalues and eigenvectors of a complex hermitian 4 x 4 matrix. The matrix is listed as the first part of the program output; the real part of each element is printed first, followed by the imaginary part. The eigenvalues are real (a property of hermitian matrices), but the eigenvectors are complex.

```
      PROGRAM COMPHER(OUTPUT)
C
C TEST PROGRAM TO TEST EIGCHER.
C AR, AI IS A 4 X 4 COMPLEX MATRIX WITH EIGENVALUES Ø.Ø, 4.Ø,
C 2. - 2. * SQRT( 2. ), 2. + 2. * SQRT( 2. )
C
      DIMENSION AR(4, 4), AI(4, 4), WORKSTG(12), EIGVAL(4),
     ,          EIGVECR(4, 4), EIGVECI(4, 4)
      DATA ((AR(I, J), J = 1, 4), I = 1, 4)
     ,                   / 3.Ø, 1.Ø, Ø.Ø, Ø.Ø,
     ,                     1.Ø, 3.Ø, Ø.Ø, Ø.Ø,
     ,                     Ø.Ø, Ø.Ø, 1.Ø, 1.Ø,
     ,                     Ø.Ø, Ø.Ø, 1.Ø, 1.Ø /
      DATA ((AI(I, J), J = 1, 4), I = 1, 4)
     ,                   /  Ø.Ø,  Ø.Ø,  Ø.Ø,  2.Ø,
     ,                      Ø.Ø,  Ø.Ø, -2.Ø,  Ø.Ø,
     ,                      Ø.Ø,  2.Ø,  Ø.Ø,  Ø.Ø,
     ,                     -2.Ø,  Ø.Ø,  Ø.Ø,  Ø.Ø /
      DATA N /4/, IORDER /4/, NVECS /1/
C
      PRINT 9ØØØ, ((AR(I, J), AI(I, J), J = 1, IORDER), I = 1, IORDER)
9ØØØ  FORMAT(1H1, 24X, "DATA MATRIX", //, (1X, 4(F5.2, 2X, F5.2, 4X)))
C
      CALL EIGCHER( N, IORDER, AR, AI, NVECS, WORKSTG, NUM, EIGVAL,
     ,                                               EIGVECR, EIGVECI )
C
      IF ( NUM .NE. IORDER ) STOP "FAILURE IN EIGCHER"
      PRINT 9Ø1Ø, (EIGVAL(I), I = 1, IORDER)
9Ø1Ø  FORMAT(1X, //, 1X, "EIGENVALUES", //, 1X, 4(4X, F7.4, 7X), /)
      PRINT 9Ø2Ø, ((EIGVECR(I, J), EIGVECI(I, J), J = 1, IORDER), I = 1,
     ,                                                          IORDER)
9Ø2Ø  FORMAT(1X, "EIGENVECTORS", //, (1X, 4(F7.4, 1X, F7.4, 3X)))
      END
```

```
                              DATA MATRIX
  3.ØØ    Ø.ØØ      1.ØØ    Ø.ØØ      Ø.ØØ    Ø.ØØ      Ø.ØØ    2.ØØ
  1.ØØ    Ø.ØØ      3.ØØ    Ø.ØØ      Ø.ØØ   -2.ØØ      Ø.ØØ    Ø.ØØ
  Ø.ØØ    Ø.ØØ      Ø.ØØ    2.ØØ      1.ØØ    Ø.ØØ      1.ØØ    Ø.ØØ
  Ø.ØØ   -2.ØØ      Ø.ØØ    Ø.ØØ      1.ØØ    Ø.ØØ      1.ØØ    Ø.ØØ

EIGENVALUES
     -.8284              -.ØØØØ              4.ØØØØ              4.8284
EIGENVECTORS
 Ø.ØØØØ  -.2706     Ø.ØØØØ   .5ØØØ     Ø.ØØØØ  -.5ØØØ     Ø.ØØØØ   .6533
 Ø.ØØØØ  -.2706     Ø.ØØØØ  -.5ØØØ     Ø.ØØØØ   .5ØØØ     Ø.ØØØØ   .6533
 -.6533  Ø.ØØØØ     -.5ØØØ  Ø.ØØØØ     -.5ØØØ  Ø.ØØØØ     -.27Ø6  Ø.ØØØØ
  .6533  Ø.ØØØØ     -.5ØØØ  Ø.ØØØØ     -.5ØØØ  Ø.ØØØØ      .27Ø6  Ø.ØØØØ
```

AUTHORS C.H. Kerr and J.M. Blair DATE March 1975

TITLE — Calculation of all eigenvalues and optionally, eigenvectors of a complex general (i.e. non-Hermitian) matrix.

ENTRY — CALL EIGCGEN(N,IORDER,AR,AI,NVECS,WORKSTG,NUM,EIGVALR, EIGVALI,EIGVECR,EIGVECI)

N — Integer input variable, the row dimension of arrays AR, AI, EIGVECR, and EIGVECI, as declared in the calling program dimension statement.

IORDER — Integer input variable, the order of the submatrix of (AR,AI) for which the eigensystem is desired. IORDER must be no greater than N.

AR,AI — Real input IORDER x IORDER arrays containing the real and imaginary parts, respectively, of the complex general matrix whose eigensystem is to be computed. The contents of AR and AI are destroyed by the routine.

NVECS — Integer input variable, a flag to indicate whether or not eigenvectors are also desired.

= 0 if only eigenvalues are to be found.
≠ 0 if eigenvectors corresponding to all eigenvalues should also be calculated.

WORKSTG — Real; one-dimensional array of size 2*N needed for working storage.

NUM — Integer output variable to return the number of eigenvalues (and eigenvectors) that were calculated.

If 0 < NUM < IORDER, the (NUM+1)th eigenvalue was not obtained after 30 iterations. Those eigenvalues that were calculated will be in the 1st NUM positions of (EIGVALR, EIGVALI). Similarly, if requested, the 1st NUM eigenvectors will be found in the 1st NUM columns of (EIGVECR, EIGVECI).

If NUM=IORDER, all eigenvalues of (AR,AI) are found in (EIGVALR,EIGVALI).

EIGVALR EIGVALI	Real output one-dimensional arrays of at least IORDER elements, containing the real and imaginary parts respectively of the eigenvalues of (AR,AI) in ascending order of modulus.
EIGVECR EIGVECI	Real output IORDER x IORDER arrays containing unnormalized eigenvectors of (AR,AI) stored by columns; i.e. the jth column of (EIGVECR,EIGVECI) contains the eigenvector corresponding to the jth eigenvalue of (EIGVALR,EIGVALI). Note that the eigenvectors are unnormalized, and each may be multiplied by an arbitrary complex number. If only eigenvalues are desired (NVECS=0), these two parameters can be omitted from the calling statement.

ROUTINES CALLED

CBABK2,CBAL,COMHES,COMLR,COMLR2, and SORT, eigenanalysis utility routines loaded from AELIB.

STORAGE

4100_8

EXIT

NUM returns the number of eigenvalues that were each calculated in less than 30 iterations. The eigenvalues of the complex general input matrix are complex; their real and imaginary parts are returned in the first NUM positions of EIGVALR and EIGVALI, respectively, in ascending order of modulus, i.e.

$$\sqrt{\text{EIGVALR(I)}^2 + \text{EIGVALI(I)}^2}$$

The corresponding eigenvectors are also complex. If requested (NVECS≠0) their real and imaginary parts are returned in the first NUM columns of EIGVECR and EIGVECI, respectively.

NORMAL EXIT: NUM=IORDER
ERROR EXIT: 0 < NUM < IORDER

One of the following error messages appears on the output file:

"ONLY NUM EIGENVALUES WERE FOUND"

or, if NVECS≠0,

"ONLY NUM EIGENVALUES AND EIGENVECTORS WERE FOUND"

ACCURACY

13 significant figures

SPEED

The following results represent average times, in seconds, of five trials, using as input matrices of random numbers from a uniform distribution on (-1,1).

	IORDER				
CALCULATION	20	40	60	80	100
ALL EIGENVALUES	.298	1.9504	5.8624	13.3518	24.9234
ALL EIGENVALUES AND EIGENVECTORS	.5662	3.957	12.4622	28.7134	54.7752

FTN OPT=2, 6600, March 1975

Note that, in general, execution time is dependent on the nature of the input matrix and its eigenvalues.

PROGRAM EXAMPLE

The program computes the eigenvalues and eigenvectors of a complex 4 x 4 matrix. The matrix is listed as the first part of the program output; the real part of each element is printed first, followed by the imaginary part. The eigenvalues and eigenvectors are complex.

```
      PROGRAM COMPGEN(OUTPUT)
C
C TEST PROGRAM TO TEST EIGCGEN.
C AR, AI IS A 4 X 4 COMPLEX MATRIX WITH EIGENVALUES 1. + 5. I,
C 2. + 6. I, 3. + 7. I, 4. + 8. I
C
      DIMENSION AR(4, 4), AI(4, 4), WORKSTG(8), EIGVALR(4), EIGVALI(4),
     ,          EIGVECR(4, 4), EIGVECI(4, 4)
      DATA ((AR(I, J), J = 1, 4), I = 1, 4)
     ,               / 5.0, 5.0, -6.0, -7.0,
     ,                 3.0, 6.0, -5.0, -6.0,
     ,                 2.0, 3.0, -1.0, -5.0,
     ,                 1.0, 2.0, -3.0,  0.0 /
      DATA ((AI(I, J), J = 1, 4), I = 1, 4)
     ,               / 9.0,  5.0, -6.0, -7.0,
     ,                 3.0, 10.0, -5.0, -6.0,
     ,                 2.0,  3.0,  3.0, -5.0,
     ,                 1.0,  2.0, -3.0,  4.0 /
      DATA N /4/, IORDER /4/, NVECS /1/
C
      PRINT 9000, ((AR(I, J), AI(I, J), J = 1, IORDER), I = 1, IORDER)
9000  FORMAT(1H1, 24X, "DATA MATRIX", //, (1X, 4(F5.2, 2X, F5.2, 4X)))
C
      CALL EIGCGEN( N, IORDER, AR, AI, NVECS, WORKSTG, NUM, EIGVALR,
     ,                                       EIGVALI, EIGVECR, EIGVECI )
C
      IF ( NUM .NE. IORDER ) STOP "FAILURE IN EIGCGEN"
      PRINT 9010, (EIGVALR(I), EIGVALI(I), I = 1, IORDER)
9010  FORMAT(1X, //, 1X, "EIGENVALUES", //, 1X, 4(F7.4, 1X, F7.4, 3X),/)
      PRINT 9020, ((EIGVECR(I, J), EIGVECI(I, J), J = 1, IORDER), I = 1,
     ,                                                           IORDER)
9020  FORMAT(1X, "EIGENVECTORS", //, (1X, 4(F7.4, 1X, F7.4, 3X)))
      END
```

```
                        DATA MATRIX

 5.00   9.00     5.00   5.00    -6.00  -6.00    -7.00  -7.00
 3.00   3.00     6.00  10.00    -5.00  -5.00    -6.00  -6.00
 2.00   2.00     3.00   3.00    -1.00   3.00    -5.00  -5.00
 1.00   1.00     2.00   2.00    -3.00  -3.00     0.00   4.00


EIGENVALUES

 1.0000  5.0000    2.0000  6.0000    3.0000  7.0000    4.0000  8.0000

EIGENVECTORS

 2.0000  -.0000     .5000   .0000    2.0000   .0000    2.0000   .0000
 1.0000  0.0000    1.0000  0.0000    2.0000   .0000    2.0000   .0000
 1.0000  -.0000     .5000   .0000     .0000   .0000    2.0000   .0000
 1.0000   .0000     .5000  -.0000    2.0000   .0000    -.0000   .0000
```

AUTHORS C.H. Kerr and J.M. Blair DATE March 1975

1-10 SOLUTION OF NONLINEAR EQUATIONS AND FUNCTION EXTREMA

00-09 ZEROES OF A POLYNOMIAL

1-10-00	CUBIC	ROOTS OF A CUBIC EQUATION
1-10-01	QUARTIC	ROOTS OF A QUARTIC EQUATION

10-19 ZEROES OR EXTREMA OF ONE FUNCTION OF ONE VARIABLE

1-10-10	ZERO	ZEROES OF F(X) USING AITKEN INTERP.
1-10-11	ZEROM	ZEROES OF F(X) USING MUELLERS METHOD
1-10-12	STEPMIN	MINIMUM OR MAXIMUM OF F(X)

30-39 M FUNCTIONS OF N VARIABLES M>=N

1-10-31	NLSPAR	SOLVE A SPARSE NONLINEAR SYSTEM

1-10

1-10 SOLUTION OF NONLINEAR EQUATIONS AND FUNCTION EXTREMA

Introduction

This section of AELIB contains the following subroutines:

CUBIC to find the zeros of a cubic polynomial with real coefficients

QUARTIC to find the zeros of a quartic polynomial with real coefficients

ZERO, ZEROM to find a zero of a function of one variable

STEPMIN to find a minimum (or a maximum) of a function of one variable

NLSPAR to find a solution of a system of m nonlinear equations with n unknowns ($m \geq n > 1$)

Also, the IMSL library [1] has a chapter on zeros and extrema which includes the following useful routines:

ZCPOLY to find the zeros of a polynomial with complex coefficients

ZQADC to find the zeros of a quadratic function with complex coefficients

ZX1LP, ZX2LP, ZX3LP linear programming

For nonlinear equations and nonlinear least-squares fitting problems, some excellent routines are available in the MINPACK package [2] (see recommendations below); also, users should note the AELIB routines NL2SOL/NL2SNO and NL2INT described in Section 1-11.

Recommendations

(a) Zeros and extrema of a function of one variable.

The AELIB routines are recommended as all known problems with these routines have been removed. When the AELIB routines are not applicable, a user should use the appropriate routine in the IMSL library.

(b) Solution of n nonlinear equations with n unknowns.

The following MINPACK routines are recommended:

HYBRJ1 requires the user to provide the Jacobian matrix (i.e. the matrix containing the partial derivatives of the functions with respect to the unknowns).

HYBRD1 does not require a user to provide the Jacobian matrix.

(c) Solution of m nonlinear equations with n unknowns (m>n) and nonlinear least squares fitting problems.

The solution here is in the least squares sense, that is, a solution which minimizes the least squares function of the m nonlinear functions. All routines recommended here use the quadratic model to approximate the least squares function, and solves the minimization problem by the Levenberg-Marquardt (L-M) method (as in the MINPACK routines) or a variant of the L-M method (as in NL2SOL/NL2SNO and NL2INT). The success of these methods depends strongly on the properties of the least squares function and the initial estimate of the solution that a user has to provide. Frequently, a user may need to try many times (with widely different initial estimates if one has no idea where a solution is) to obtain a solution. We recommend the following routines.

c.1 Solution of m nonlinear equations with n unknowns (m>n) or nonlinear least squares fitting problem with equal weighting.

LMDER1, NL2SOL: requires a user to provide the Jacobian matrix.

LMDIF1, NL2SNO: does not require a user to provide the Jacobian matrix.

NLSPAR: use this AELIB routine when the system of equations is too large to be treated by the above routines.

Notes: 1. LMDER1 and LMDIF1 are MINPACK routines. Information about the MINPACK package is available in the system bulletin SYSBULL,MINPACK.

2. For least squares fitting problems, the n parameters to be fitted are the n unknowns, and the residue functions, one at each of the m data points, are the m nonlinear functions.

That is,

$$R_i = y_i - f(X_i;P_j), \ i=1,2,\ldots,m; \ j=1,2,\ldots n$$

where R_i are the m residue functions and P_j the parameters to be fitted.

c.2 Nonlinear least squares fitting with unequal weighting.

NL2INT: AELIB routine described in Section 1-11.

Methods

No attempt is made to describe the methods here. Ref [3] is an excellent reference on the L-M method for nonlinear least-squares optimization as well as other methods for optimization or solution of a function of one or more variables.

Peter Wong
1983 May 31

1-10 SOLUTION OF NONLINEAR EQUATIONS AND FUNCTION EXTREMA

References

[1] IMSL Library Reference Manual.

[2] User Guide for MINPACK-1 by J.J. Moré, B.S. Garbow and K.E. Hillstrom, ANL-80-74, August 1980.

[3] Practical Optimization by P.E. Gill, W. Murray and M. Wright, Academic Press, 1981.

TITLE Solve a cubic equation of the form $ax^3+bx^2+cx+d=0$

ENTRY CALL CUBIC(AA,RRE,RIM)

AA real one-dimensional input array of length four supplying the coefficients a, b, c and d of the equation to be solved
AA(1)=a, AA(2)=b, AA(3)=c, AA(4)=d

$\left.\begin{array}{l}\text{RRE}\\\text{RIM}\end{array}\right\}$ real one-dimensional output arrays of length three giving the real and imaginary parts of the three roots. The first root is always real.

EXIT The real parts of the roots are returned in RRE and the imaginary parts in RIM. No error checking is done.

SPEED ∿0.3 ms (6600, February 1969)

ACCURACY ∿13 significant figures

REFERENCE Handbook of Engineering Fundamentals, Eshbach, Section 2-13.

AUTHOR Translated from APEX C-3-4 by C.J. Johnson Feb. 1969

TITLE Solve a quartic equation of the form $ax^4+bx^3+cx^2+dx+e=0$

ENTRY CALL QUARTIC(AA,RRE,RIM)

AA real one-dimensional input array of length five supplying the coefficients a, b, c, d and e of the equation to be solved.

AA(1)=a, AA(2)=b, AA(3)=c, AA(4)=d, AA(5)=e

$\left.\begin{array}{l}\text{RRE}\\\text{RIM}\end{array}\right\}$ real one-dimensional output arrays of length four giving the real and imaginary parts of the four roots.

EXIT The real parts of the roots are returned in RRE and the imaginary parts in RIM.

If e=0, a failure exit is made and "ILLEGAL COEFFICIENT IN QUARTIC" is printed. In this case, the contents of RRE and RIM are unpredictable.

ACCURACY In most cases, approximately 13 figures are correct.

However, in some cases of equal roots, only about eight figures can be correctly calculated. In these cases, "QUARTIC DID NOT OBTAIN FULL ACCURACY" is printed and these less accurate results are returned in RRE and RIM.

SPEED Approximately .97 ms (6600, March 1969)

AUTHOR C.J. Johnson DATE March 1969

TITLE Zero of a Function f(x)

ENTRY R = ZERO(A,AUX,H,E,ERROR)

A real input variable, the initial estimate of the root.

AUX real function subprogram supplied by the user to evaluate f(x). It has one formal parameter X.

H real input variable, the size and direction of the step for the routine to take from A to find the area of the true root.

E real input variable, the relative error required for convergence of ZERO. If x_i, x_{i+1} are successive approximations to the root such that $|x_{i+1}-x_i| \leq |h.e|$, then x_{i+1} is returned as the zero of f(x).

ERROR subroutine supplied by the user to handle the error exits from ZERO. In the event of an error it will be called from ZERO with its one formal parameter y set to 1., 2., or 3. depending on the type of error encountered.

The calling program must contain the statement EXTERNAL AUX,ERROR.

ROUTINES CALLED AUX,ERROR supplied by the user.

EXIT If no errors are detected, the real value of the root is returned to the calling routine in R.

If there is an error, the routine exits via SUBROUTINE ERROR (y) with:

y=1., ZERO has not found the general area of the root after stepping 20H from A

A return from ERROR causes ZERO to search for another 20 steps. If the root area still cannot be found, ZERO exits with the root = 0.

y=2., $f_{i-1} - f_i = 0$. This error may occur if E is extremely small.

A return from ERROR produces a successful exit.

y=3., during the process of refining the root, the required accuracy has not been found in 10 cycles.

A return from ERROR causes further refining for 10 cycles. If the required accuracy still cannot be met, ZERO exits with the root = 0.

SPEED This depends on the speed of AUX. The number of times AUX is used depends on A, H and E.

ACCURACY This depends on the product of H and E and also on the accuracy to which AUX is calculated.

AUTHOR D.C. Knowles DATE March 1969

TITLE

Zero(s) of a Function Using Mueller's Iterative Method for Real Odd Order Roots (single, triple, etc.) and STEPMIN Function for Real Even Order Roots (double, quartic, etc.)

ENTRY

R = ZEROM(A,AUX,H,E,IFLAG)

A — real input variable, the initial estimate of the root.

AUX — real function subprogram supplied by the user to evaluate f(x). It has one formal parameter, X, and must be declared EXTERNAL in the calling program.

H — real input variable, the size and direction of the step for the routine to take from A to find the real root.

E — real input variable, specifying the upper bound of the relative error of the root.

IFLAG — integer output variable, a flag specifying the type of zero found or whether an error was encountered as defined in EXIT section.

ROUTINES CALLED

STEPMIN from AELIB, AUX supplied by the user.

COMMON BLOCKS USED

LASTV

EXIT

The function ZEROM always returns a value, but IFLAG must be tested to get the correct interpretation.

IFLAG=0 — function value is a real odd order root (single, triple, etc.).

IFLAG=1 — function value is a real even order root (double, quartic, etc.).

IFLAG=2 — function value is indefinite because ZEROM has not found a change of sign of the function after stepping 20H from A and a double root has not been found.

IFLAG≥3 function value is indefinite because ZEROM has not converged on a single root. IFLAG=4 means that the X iterates are within relative error, E, but the function values are not within 100*E. IFLAG=3 indicates general nonconvergence.

The common block LASTV is defined as follows:

```
COMMON LASTV/X1,X2,FX1,FX2/
```

If IFLAG is 0 or 1, X1 is the root and FX1 is the "zero" function value.

If IFLAG is 3 or 4, X1 and X2 define the interval in which a root is suspected. FX1(FX2) is the function value at X1(X2).

SPEED This depends on several factors: the speed of AUX, how good the initial approximation is, whether a double root is indicated and whether a search for it is attempted.

ACCURACY This depends on the relative error, E, and the accuracy to which AUX is calculated.

REFERENCE System 1360 Scientific Subroutine Package (360A-CM-03X) Version III Programers Manual, International Business Machines Corporation, 1968, pages 217-219.

AUTHOR L.E. Evans DATE October 1973

TITLE Minimize (Maximize) a Function of a Single Variable F(X).

ENTRY R = STEPMIN(AUX,XO,D,E,IST,NO,ISGN)

AUX real function subprogram, supplied by the user to evaluate F(X). It has one formal parameter X, and the calling program must contain the statement "EXTERNAL AUX".

XO real input variable, an initial approximation to the minimum.

D real input variable, the step size of X used to reach the minimum (maximum) region. The routine expects that for a given D and NO (see below), the minimum (maximum) lies within the interval (XO-NO*D, XO+NO*D).

E real input variable, the absolute accuracy to which the minimum is required.

IST integer output variable, to return status of the result (see EXIT).

NO integer input variable, the maximum number of iterations required to reach the minimum (maximum) region. NO ≤ 100.

ISGN integer input variable, used as a flag as follows:
=1 find minimum
=-1 find maximum

ROUTINES CALLED AUX, supplied by the user.

EXIT This function returns in R the value of X at which the minimum (or maximum) of the funotion f(X) occurs. XO is changed only if no minimum has been found; in that case the function returns an indefinite value and XO takes the last value reached in the search.

Information on the type of minimum found is returned in IST:

IST=1	normal termination
IST=2	F(X)=F(X-E)(X being the minimum)
IST=3	F(X)=F(X+E)
IST=4	F(X-E)=F(X)=F(X+E)
IST=5	F(X-D)<F(X)
IST=6	F(X+D)<F(X)

IST=5 or 6 occurs only if no minimum was found after NO iterations.

SPEED

Time depends on the speed of AUX.

Number of calls to AUX varies with the combinations of X and D for the first part of the iterations. The maximum number of calls is NO+3 since it takes three calls to set up the search.

In the second part, convergence takes N steps; N being the smallest integer $\geq \log_2(E/D)$.

Since there are two calls to AUX every step, there will be exactly 2N calls to AUX in this final part.

EXAMPLE

The following program uses STEPMIN to find the minimum of the function $y = (x-2)^2-2$. The auxiliary routine and program results are also provided.

```
      PROGRAM TEST(OUTPUT)
      EXTERNAL AUX
C
C     TO FIND THE MINIMUM OF Y = (X-2)**2-2
C
      X0=1.
      D=10.
      E=0.0001
      NO=20
      ISGN=1
      XMIN=STEPMIN(AUX,X0,D,E,IST,NO,ISGN)
      PRINT 1
      PRINT 2,X0,D,E,NO,ISGN
      PRINT 3,IST,XMIN
    1 FORMAT(1H1//5X,*STEPMIN TEST SAMPLE*/5X,19(*-*)//)
    2 FORMAT(2X,*X0 = *,1PE12.5/
     12X,*D = *,1PE12.5/
     22X,*E = *,1PE12.5/
     32X,*NO = *,I2/
     42X,*IGSN = *,I2///)
    3 FORMAT(2X,*OUTPUT RESULTS*/2X,13(*-*)//
     15X,*IST = *,I2/5X,*XMIN = *,1PE12.5)
      END
```

```
FUNCTION AUX(X)
AUX=X*X-4.*X+2.
END
```

```
   STEPMIN TEST SAMPLE
   -------------------

X0 =  1.00000E+00
D =  1.00000E+01
E =  1.00000E-04
NO = 20
IGSN =  1


OUTPUT RESULTS
-------------

   IST =  1
   XMIN =  1.99998E+00
```

AUTHOR A. Perreault and J.M. Blair DATE August 1973

DOCUMENTATION
REVISED BY P.Y. Wong DATE February 1979

TITLE

NLSPAR: Solve a Sparse System of Nonlinear Equations

$$\tau_i(x_1,x_2,\ldots,x_N) = 0,\ i=1,\ M \text{ for } M \geq N$$

and Perform Subsequent Statistical Analysis.

INTRODUCTION

τ_i consists of nonlinear components $f_i(x)$ and linear components $a_{ij}x_j$ which can be expressed

$$\tau_i(x_1,x_2,\ldots,x_N) = f_i(x_1,x_2,\ldots,x_N) + \sum_{j=1}^{N} a_{ij}x_j,\ i=1,M$$

The linear components may be absent, in which case A would be a null matrix.

The solution requires the M by N Jacobian matrix J of the nonlinear component. The elements of the J matrix, $j_{ij} = \partial f_i/\partial x_j$ may be specified analytically by the user or computed by finite differences within NLSPAR.

As the equations are nonlinear, the solution is not always exact and is a solution in the least squares sense in that the sum of the squares of residuals $\Sigma\tau_i^2$ is minimized. After the initial solution has been performed, the routine may be re-entered to obtain the variance-covariance matrix.

ENTRY

CALL NLSPAR(FUNC,M,N,X,SAC,STPMIN,MAXFUN,W,IW,IRN,IP,A, IRNA,IPA,HMAX,IE)

FUNC Name of a subroutine written by the user which has the form

```
SUBROUTINE FUNC(M,N,X,F,D)
```

It must calculate $f_i(x_1,x_2,\ldots,x_N)$, $i=1,2,\ldots,M$ and store them in array F. If the non-zero derivatives $\partial f_i/\partial x_j$ are to be specified analytically they should be stored by columns, j, in array D; otherwise D is a dummy argument and derivatives $\partial f_i/\partial x_j$ are calculated within NLSPAR by finite difference approximation. The statement "EXTERNAL FUNC" must appear in the calling program.

M Integer input variable specifying the number of equations.

N Integer input variable specifying the number of independent variables x_j. $N \leq M$.

X Real array, dimension $\geq N$, to be set by the user to an approximate solution or starting value assumption. On return it contains the best solution found by NLSPAR.

SAC Real variable set by the user to an acceptable value for the sum of the squares of residuals

$$S = \sum_{i=1}^{M} [\tau_i(x_1, x_2, \ldots, x_N)]^2.$$

A return is made as soon as $S \leq SAC$. If SAC is set zero, return is controlled by STPMIN alone.

STPMIN Real variable set by the user to indicate the accuracy to which the user wishes to calculate $x_1, x_2, \ldots, x_n$. A return is made if an iteration makes changes δ_i to x_i such that $||\delta|| < STPMIN$ where

$$||\delta||^2 = \sum_{i=1}^{N} \delta_i^2.$$

The user may also set STPMIN to zero since a return is also made if $||\delta|| < EPS * ||x||$ where EPS is the relative machine accuracy

MAXFUN Integer variable set by the user to the maximum number of calls of FUNC allowed. An error return is made if a solution has not been found within this number of calls.

W Real array, dimension IW, used for workspace, during NLSPARL and to return results to the calling routine as defined in EXIT section. $IW \geq 4*M+7*N+3*M*N+10$.

IW Integer input variable specifying the dimension of the array W.

IRN and IP Integer arrays, dimensions NIRN and NIP respectively, to describe the sparsity structure of the Jacobian, J. If the Jacobian is to be calculated in NLSPAR by finite differences, IRN and IP are also set internally.

However, if the Jacobian elements are to be specified analytically by the user, the Jacobian structure must first be specified in IRN and IP which define the location of non-zero terms. The non-zero derivatives are stored by columns; e.g.

$$\partial f_1/\partial x_1,\ \partial f_3/\partial x_1,\ \partial f_6/\partial x_1,\ \ldots,\partial f_2/\partial x_2,\ \partial f_4/\partial x_2,\ \ldots$$

IP(J) points to the position of the beginning of the Jth column in this ordering, unless this column is null, in which case IP(J) equals IP(J+1) and IRN(K) indicates the row number of the Kth non-zero. IP(N+1)-1 equals the number of non-zero derivatives. NIRN $\geq$ the number of non-zero derivatives and NIP $\geq$ N+1.

A Real array set by the user to contain the non-zero coefficients a_{ij}, stored by columns; e.g.

$$a_{11},\ a_{31},\ a_{51},\ a_{22},\ a_{42},\ a_{13},\ a_{53},\ a_{34},\ a_{55}.$$

Dimension must be at least NIRNA below.

IRNA and IPA Integer arrays of dimension NIRNA and NIPA respectively. These are set to index a_{ij} in the same way as IRN and IP index $\partial f_i/\partial x_j$. The special case where $a_{ij}=0$ for all i and j may be indicated by setting IPA(1)=0. (For the matrix A above IRNA=1,3,5,2,4,1,5,3,5 and IPA=1,4,6,8,9,10.)

HMAX — Real input variable ≥ 0. Set HMAX to zero if derivatives are to be supplied analytically through FUNC, or to an upper limit for step-lengths Δx to be used when differencing to approximate derivatives by $J_{ij} \approx \Delta f_i/\Delta x_j$.

IE — Integer input variable set to zero if the user wishes to solve a sparse system of non-linear equations.

If the user wishes to generate a column of the variance-covariance matrix after a successful call to NLSPAR with IE=0, NLSPAR is called again with IE set to the column required. On the next exit, the X array will then contain the IEth column of the variance covariance matrix.

ADDITIONAL ENTRY INFO — The subroutine also contains a common block

```
COMMON/SPARS/LP,EPS,ISCALE,IPRINT
```

These variables are set to 6, 1.0E-14, 0, 0 respectively. If the user wishes to change one of these, he must include the entire common block in his calling program.

LP — Integer variable specifying the unit number for printing of any results requested by the IPRINT parameter.

EPS — Real variable, the relative machine accuracy (round-off error).

ISCALE — Integer variable, used as an interface parameter and must be included if the common block is used in routine FUNC, but should not be set.

IPRINT — Integer variable controlling printing. IPRINT=0 gives no printing. IPRINT≠0 results in printing at the first and last iteration and at every IABS(IPRINT)th iteration in between. If IPRINT>0, then the iteration number, the Marquart parameter (see reference), the sum of squares of residuals S, the number of calls of FUNC made so far, and the Euclidean norm

$$||x||^2 = \sum_{i=1}^{N} x_1^2$$

of the current approximate solution, the norm of the last change to the solution and the norm of the vector

$$v = \{\partial\tau_j/\partial x_j\}^T\tau$$

(which is the gradient of 1/2 S) are printed. If IPRINT<0, then the current solution x, current residual vector τ and the current gradient vector v are also printed.

EXIT

Normal Exit: Control is returned to the calling program with the best solution found in the x array, the current residual vector τ in W(I), I=1, ...,M; the current vector f (if the $\{a_{ij}\}$ matrix is non zero) in W(I+M), I=1, ...,M; the current matrix of derivatives $\{\partial f_i/\partial x_j\}$ or its approximation stored by columns in the form specified by IRN and IP in W(NPHI+I), I=1,2,..., where NPHI=M if $\{a_{ij}\}$ is a zero matrix, or 2M if not; the number of calls of FUNC made in W(IW-1); and the length of working storage actually used in W(IW).

Error Exit:

(1) If workspace W is not large enough, the message

***WORKSPACE W IS TOO SMALL

is printed, W(IW-1) is set to zero, and control is returned to the calling routine.

(2) If more than MAXFUN calls for FUNC are required, the message

***MORE THAN MAXFUN CALLS OF FUNCTION NEEDED

is printed, MAXFUN is set to 1, and control is returned to the calling routine.

(3) If NLSPAR is called with IE=∅ without a previous call with IE=0, the message

***CALL OF NLSPAR WITH IE=0 DOES NOT PRECEDE THE IE≠0 CALL

is printed and control is returned to the calling routine.

(4) Overpopulation of matrices may cause the indices used by NLSPAR to exceed 2^{17} which is not allowable in current FORTRAN. In this case, the message

***INTEGER WORD LENGTH LIMITATION RENDERS THIS PROBLEM UNSOLVABLE

is printed and control is returned to the calling routine.

ROUTINES CALLED

User Supplied

FUNC calculates F (and D when HMAX=0).

AELIB Routines Used

JACOB calculates [J] when HMAX=∅.

SORTAG sorts and tags an array.

PDSPARL solution of linear equations in positive definite symmetric sparse matrix.

Utility Routines Loaded with NLSPAR

NLPLIN calculates $[Y] + [A][X]$ or $[Y] + [A]^T[X]$.

SUMSQ calculates $S = \sum_{i=1}^{n} a_i b_i$

SUBNL performs initialization for solution of linearized minimization problems.

COMMON BLOCKS USED SPARS, NLIN, NLSNL

STORAGE 3700_8

ACCURACY Specified by the users in the given value of SAC and STPMIN.

SPEED Depends on the complexity of the function and the required accuracy. The time the example program USE1 takes to execute is 0.104 s. The example program USE2 (using the analytic derivatives supplied) takes 0.091 s. (6600, July 1975)

EXAMPLES Consider the following non-linear equations where m=5.

$$\tau_1 = -(3 - 0.5x_1)x_1 + 2x_2 - 1,$$

$$\tau_i = x_{i-1} - (3 - 0.5x_i)x_i + 2x_{i+1} - 1, \quad i=2,3,\ldots,M-1,$$

$$\tau_m = x_{m-1} - (3 - 0.5x_m)x_m - 1.$$

These equations could be programmed straight into the FUNC subroutine and the matrix, A, would be zero (i.e. IPA(1)=0), however, for the sake of accuracy (and the examples) the matrix, A, will be used, linear terms are parenthesized.

$$\tau_1 = [-3x_1 + 2x_2] + 0.5x_1^2 - 1,$$

$$\tau_i = [x_{i-1} - 3x_i + 2x_{i+1}] + 0.5x_1^2 - 1, \quad i=2,3,\ldots,M-1.,$$

$$\tau_m = [x_{m-1} - 3x_m] + 0.5x_m^2 - 1.$$

Therefore,

$$[a] = \begin{bmatrix} -3 & 2 & 0 & 0 & 0 \\ 1 & -3 & 2 & 0 & 0 \\ 0 & 1 & -3 & 2 & 0 \\ 0 & 0 & 1 & -3 & 2 \\ 0 & 0 & 0 & 1 & -3 \end{bmatrix}$$

and the functions are $f_i = 0.5\ x_1^2 - 1$, $i=1,2,\ldots,M$.
$\partial f_i/\partial x_j = x_i$ if $j=i$; 0 if $j \neq i$.

PROGRAM EXAMPLES

Two example programs are provided: The first, USE1, is the case where the derivatives are supplied analytically in FUNC. The second, USE2, has the derivatives approximated internally.

To set the A array and its associated pointers, IRNA and IPA, the user can either set them directly as in USE1, or call the AELIB routine PACKER, passing the A matrix as a parameter to set the arrays as in USE2.

```
      PROGRAM USE1(OUTPUT,TAPE6=OUTPUT)
      INTEGER IP(6),IRN(14),IPA(6),IRNA(14)
      REAL W(135),A(14),X(5)
      EXTERNAL FUNC
C
C     INITIALIZATION
C
      M=5
      N=5
      SAC=1.ØE-18
      STPMIN=1.ØE-1Ø
      MAXFUN=2ØØ
      IW=135
      HMAX=Ø.Ø
      IE=Ø
C
C     SET UP A AND ITS INDEX ARRAYS, IRNA AND IPA
C
      A(1)=A(4)=A(7)=A(1Ø)=A(13)=-3.Ø
      A(2)=A(5)=A(8)=A(11)=1.Ø
      A(3)=A(6)=A(9)=A(12)=2.Ø
      IRNA(1)=IRNA(3)=1
      IRNA(2)=IRNA(4)=IRNA(6)=2
      IRNA(5)=IRNA(7)=IRNA(9)=3
      IRNA(8)=IRNA(1Ø)=IRNA(12)=4
      IRNA(11)=IRNA(13)=5
      IPA(1)=1
      IPA(2)=3
      IPA(3)=6
      IPA(4)=9
      IPA(5)=12
      IPA(6)=14
C
C     SET THE POINTER ARRAYS FOR THE JACOBIAN MATRIX
C     AND INITIALIZE X.
C
      DO 1Ø I=1,5
         X(I)=-1.Ø
         IP(I)=I
         IRN(I)=I
   1Ø CONTINUE
      IP(6)=6
C
```

```
      CALL NLSPAR(FUNC,M,N,X,SAC,STPMIN,MAXFUN,W,IW,IRN,IP,A,
     ,IRNA,IPA,HMAX,IE)
      PRINT 7,(X(I),I=1,5)
    7 FORMAT(1H1,*SOLUTION FROM NLSPAR IS*,/,2X,*(*,4(F9.6,*,*),F9.6,*)*
     ,)
C
      END

      SUBROUTINE FUNC(M,N,X,F,D)
      DIMENSION X(N),F(M),D(M)
C
      DO 1Ø I=1,M
         F(I)=Ø.5 * X(I)**2 - 1.Ø
         D(I)=X(I)
   1Ø CONTINUE
      RETURN
      END
```

```
SOLUTION FROM NLSPAR IS
 ( -.968354,-1.186958,-1.148478, -.958989, -.594159)
```

```
      PROGRAM USE2(OUTPUT,TAPE6=OUTPUT)
      INTEGER IP(6),IRN(14),IPA(6),IRNA(14)

      REAL W(135),A(14),X(5),AMATRIX(5,5)
      EXTERNAL FUNC

      DATA AMATRIX/-3.Ø,1.Ø,3*Ø.Ø,2.Ø,-3.Ø,1.Ø,3*Ø.Ø,2.Ø,-3.Ø,1.Ø,
     ,             3*Ø.Ø,2.Ø,-3.Ø,1.Ø,3*Ø.Ø,2.Ø,-3.Ø/
C
C     INITIALIZATION
C
      M=5
      N=5
      X(1)=X(2)=X(3)=-1.Ø
      X(4)=X(5)=-.1Ø
      SAC=1.ØE-18
      STPMIN=1.ØE-1Ø
      MAXFUN=2ØØ
      IW=135
      HMAX=1.Ø
      IA=14
      IE=Ø
C
C     SET UP A AND ITS INDEX ARRAYS, IRNA AND IPA,FOR (A(I,J)) MATRIX.
C
      CALL PACKER(A,AMATRIX,IRNA,IPA,M,N,IA)
C
C     THE POINTER ARRAYS FOR THE JACOBIAN MATRIX WILL BE SET IN NLSPAR
C
      CALL NLSPAR(FUNC,M,N,X,SAC,STPMIN,MAXFUN,W,IW,IRN,IP,A,
     ,            IRNA,IPA,HMAX,IE)
      PRINT 7,(X(I),I=1,5)
    7 FORMAT(1H1,*SOLUTION FROM NLSPAR IS*,/,2X,*(*,4(F9.6,*,*),F9.6,*)*
     ,)
      END
      SUBROUTINE FUNC(M,N,X,F,D)
      DIMENSION X(N),F(M)
C
      DO 1Ø I=1,M
         F(I)=Ø.5 * X(I)**2 - 1.Ø
   1Ø CONTINUE
      RETURN
      END
```

```
 SOLUTION FROM NLSPAR IS
  ( -.968354,-1.186958,-1.148478, -.958989, -.594159)
```

REFERENCE J.K. Reid, "FORTRAN Subroutines for the Solution of Sparse Systems of Non-Linear Equations", AERE-R-7293.

AUTHOR Adapted by D.E. Smith and M.B. Carver from the Harwell Routines NSO3A and NSO3C DATE July 1975

1-11 FITTING USER SPECIFIED FUNCTIONS TO DATA

00-09 LINEAR REGRESSION TO A POLYNOMIAL IN X

1-11-00	POLFIT	LEAST SQUARES FIT TO A POLYNOMIAL IN X

10-19 LINEAR REGRESSION TO A POLYNOMIAL IN (X1,...XK)

1-11-10	PRFIT	FIT A POLYNOMIAL OF DEG I IN (X1,...XK)
1-11-11	POLREG	FIT A GIVEN POLYNOMIAL IN (X1,...XK)
1-11-12	COMKTM	DEFINE A POLYNOMIAL OF DEG I IN (X1,..XK
1-11-13	SSCP	TRANSFORM FITTING DATA FOR RLSTEP
1-11-14	PCALC	COMPLETE FIT DONE BY RLSTEP
1-11-15	PSTATF	CALCULATE STATISTICS FOR REGRESSION
1-11-16	CIPLOT	PLOTTING CONFIDENCE INTERVALS

20-29 NON-LINEAR LEAST SQUARES FIT TO A USER"S FUNCTION

1-11-20	MLSQQ	NONLINEAR LEAST SQUARES FIT TO F(X1,.XK)
1-11-21	YFST	FITTED VALUE STANDARD DEVIATION ESTIM
1-11-24	NL2INT	NL2SOL INTERFACE
1-11-25	NL2SOL	NL2SNO MINIMIZE SUM OF SQUARES OF DIFF FUNC

30-39 SPECIAL FITTING APPLICATIONS

1-11-30	RADCURV	APPROX RADIUS AND CENTRE OF CURVATURE

1-11

1-11 FITTING USER SPECIFIED FUNCTIONS TO DATA

Introduction

The subroutines in this section of AELIB perform least squares fitting (or regression) of functions to data. That is, given a function

$$R(p) = (R_1(p), R_2(p), \dots, R_n(p))^T$$

of m parameters $p = (p_1, p_2, \dots, p_m)^T$, find a parameter vector p* which minimizes the sum-of-squares function

$$F(p) = \sum_{i=1}^{n} R_i(p)^2 \qquad (1)$$

Often the function R is defined as a residual vector. In this case given a set of n data points, (X_i, y_i), i=1,...,n, and weights w_i for each y_i value, a function, f(p;X) is fitted to the data which minimizes the sum of squares,

$$F(p) = \sum_{i=1}^{n} w_i(y_i - f(p;X_i))^2 \qquad (2)$$

y is called the dependent variable and X is called the independent variable. X may in fact be a vector $(x_1, \dots, x_k)$ of variables in k dimensional space.

Linear vs Nonlinear Regression

If R or f is linear in $p=(p_1, \dots, p_m)$, then estimating values for p is called linear regression or linear least squares fitting. If R or f is not linear in p, estimating values for p is called nonlinear regression or nonlinear least squares fitting. Linear regression is equivalent mathematically to the solution of a linear system of equations and is therefore computationally straightforward. Nonlinear regression requires an iterative solution of a nonlinear system and is much more involved. Therefore, do not use a nonlinear regression routine where a linear one will do.

Weighting

If measurements of the dependent variable y are normally distributed with mean y_i and variance v_i then the least squares solution with $w_i=1/v_i$ is the maximum likelihood solution, that is the solution which is statistically "most likely" based on the data.

Because of this, the weights used should be $w_i=(\text{variance}(y_i))^{-1}$. If such variance estimates are not known, it is common practice to assume the variance is constant and to weight all residuals equally. This is called an unweighted fit. $w_i=1$ should be used in this case because the minimum sum of squares $\sum_{i=1}^{n} w_i(y_i-f(X_i))^2$ is then used to estimate the fixed but unknown variance.

1-11 FITTING USER SPECIFIED FUNCTIONS TO DATA

Statistical Analysis

Statistical analysis of a least squares fit is a subject in itself and will not be described here. Consult a general regression text, such as [1], for this.

The AELIB subroutines PSTATF, MLSQQ, NL2INT, YFST and NL2SOL provide the following statistical data: residual (weighted) sum of squares, estimates of parameter variance and covariance (in what we call the error matrix) and estimates of the fitted function value variance.

From the variance estimates, confidence intervals may be calculated, as illustrated in the MLSQQ writeup; predicted function value variance may be estimated, as shown in the YFST writeup; or other statistical analyses can be performed.

If the fit was weighted, the residual sum of squares should be subjected to a X^2 or "goodness of fit" test.

Linear Regression Routines

Linear regression is almost always thought of in the form of (2).

While in theory we can fit any linear function we wish, the most common requirement is fitting to a polynomial in the independent variables $(x_1,\ldots,x_n)$. The AELIB routine POLFIT fits a polynomial in one independent variable using an algorithm which compensates for the ill-conditioned linear system which is characteristic of polynomial regression. No statistical data other than the residual sum of squares is provided by POLFIT. The IMSL routine RLFOR [2] is slightly more involved to use but employs the same algorithm as POLFIT and also provides extensive statistical analysis.

The AELIB routines PRFIT and POLREG fit a polynomial in several independent variables $(x_1,\ldots,x_n)$ using a stepwise regression routine, RLSTP, from IMSL. PRFIT and POLREG use only those polynomial terms that are significant at the 95% significance level so the fitted function will most likely not contain all the possible terms. Following the fit, the routine PSTATF may be called to provide statistical data.

PRFIT and POLREG are modular in form using modules COMKTM, SSCP and PCALC. These modules can be used to call RLSTP directly to force some terms into the model, to change the significance level for terms, etc.

1-11 FITTING USER SPECIFIED FUNCTIONS TO DATA

The above linear regression routines appear to fulfill the needs of our users. However, Chapter R of IMSL Library has quite extensive coverage in this area and should be consulted for other regression requirements.

Nonlinear Regression Routines

Fitting nonlinear functions to data can be an extremely difficult problem. For nonlinear least squares fitting there are many possible iterative algorithms. See [3] for a list of some of these.

The AELIB subroutines MLSQQ and NL2SOL and the IMSL subroutine ZXSSQ [2] all use iterative algorithms that solve a succession of linear least squares problems. An initial guess for the parameter values is required and from this a linear system

$$AD=B$$

where A is the m x m normal matrix,
D is the m vector of corrections to the parameters $(p_1,p_2,\ldots,p_m)$,
and
B is an m vector

is created which approximates the nonlinear problem. This system is solved for D which gives both new values for the parameters and a new linear system. The process is repeated until D is very small or some other criterion is met.

Nonlinear Regression Models

Nonlinear regression algorithms differ in the models they use to create the approximating linear system. MLSQQ and ZXSSQ use the quadratic Gauss-Newton model in which

$$A = J^T(p)J(p)$$

$$B = -J(p)R(p)$$

where J(p) is the Jacobian matrix with elements

$$j_{i,j} = \frac{\partial R_i(p)}{\partial p_j}$$

The closer the model is to the actual nonlinear problem, the faster convergence will take place since the minimum of the model will tend to coincide with p*, the minimizer of the original problem. The subroutine NL2SOL uses either the Gauss-Newton model or an augmented Gauss-Newton model, switching internally to the one that best approximates the nonlinear problem. The augmented model involves adding a correction to the matrix A so that the linear system better approximates a quadratic Taylor expansion about the current parameter values. This model is discussed in more detail in [4].

Nonlinear Regression Iteration

In solving the linear system AD=B, the AELIB subroutine MLSQQ and the IMSL subroutine ZXSSQ both use variants of the Marquardt-Levenberg (M-L) algorithm. (The AELIB routine NLSPAR also employs this technique to solve a sparse non-linear system.) This algorithm combines the two iterative techniques of steepest descent, which is slow and reliable, and Newton-Raphson (N-R) iteration, which is fast but not too reliable. A brief description follows but consult a reference such as [1] for more details.

By introducing the Marquardt parameter, λ, and solving the modified system,

$$(A+\lambda I)D=B$$

where I is the m x m identity matrix, we have, when λ is large, a reliable steepest descent solution and, when λ is small or zero, a fast N-R solution. The choice and refinement of values for λ is what distinguishes M-L methods. References for the particular algorithms used in MLSQQ and ZXSSQ are provided in the writeups for these routines.

The subroutine NL2SOL uses a Marquardt parameter only occasionally when solving the linear system. The algorithm decides whether or not it is necessary.

The correction vector D may be thought of as a step in a particular direction of a certain size to be added to the current parameters. The model used to calculate D may only approximate the nonlinear problem in a small region around the current parameter values, called a trust region. NL2SOL modifies the step D if necessary, so that the new parameters are in the indicated direction but remain within the trust region. This can speed up convergence since nonlinear problem, rather than the model, is the basis of the parameter correction.

Nonlinear Regression Routine Recommendations

For the general problem of minimizing the sum of squares function (1) NL2SOL or NL2SNO will have to be used. The only difference between these two routines is that NL2SOL requires the user to supply partial derivatives of his function while NL2SNO approximates these derivatives. If the sum of squares function has the specific form (2) (i.e. the problem is one of data fitting) any of the subroutines NL2SOL, NL2SNO, NL2INT, MLSQQ, or ZXSSQ may be suitable.

We recommend that NL2INT be used if possible. This subroutine has the same calling sequence as MLSQQ (with the minor exceptions that a larger working storage array is needed, the arrays X and E are declared differently in the auxiliary routine, and an extra COMMON block may have to be defined) but it calls NL2SOL or NL2SNO to solve the least-squares problem.

1-11 FITTING USER SPECIFIED FUNCTIONS TO DATA

The algorithm used in NL2SOL is one of the best currently known and in many cases it will converge much faster than MLSQQ. The similar calling sequence makes it easy to replace calls to MLSQQ in existing programs.

The subroutine NL2INT uses more storage than a direct call to NL2SOL would and sets many default values for the fit. NL2SOL/NL2SNO may be called directly if necessary but much more user programming is required and the references given in the routine writeup should be consulted.

MLSQQ and ZXSSQ are small subroutines and may work well for many problems. The algorithm in ZXSSQ is better than that in MLSQQ in that the sum of squares is forced to decrease at each iteration. However, ZXSSQ requires more user programming that does MLSQQ and the partial derivatives may not be supplied by the user.

Special Applications of Nonlinear Regression

The subroutine RADCURV calls NL2SOL to fit a circle to a set of points provided by the user. From this the radius and centre of curvature of the data and confidence intervals for these values are calculated.

L.E. Evans
1981 May

Revised, C.A. Wills
1982 June

References

[1] N.R. Draper and H. Smith, Applied Regression Analysis, John Wiley and Sons, Inc.

[2] IMSL Library Reference Manual.

[3] John C. Nash, An Annotated Bibliography on Methods for Non-linear Least Squares Computations Including Test Problems, Mary Nash Information Services, Vanier, Ontario.

[4] John E. Dennis, Jr., David M. Gay, and Roy E. Welsch, "An Adaptive Nonlinear Least-Squares Algorithm," TOMS, Vol. 7, No. 3, pp 348-368.

TITLE Least Squares Fitting of a Polynomial in One Independent Variable. (Fitting to a sequence of such polynomials is also possible.)

ENTRY CALL POLFIT(X,Y,W,YF,P,N,MA,E,NE,S,WS)

X(N) real input array of values of independent variable.

Y(N) real input array of values of dependent variable.

W(N) real input array of weights for dependent variable.
For a weighted fit use $w(i)=(\text{variance}(y(i)))^{-1}$.
For a unweighted fit use w(i)=1.
Setting w(j)=0 will remove the jth data point from the fit.

YF(N) real output array of fitted values of dependent variable returned from POLFIT.

P(M) real output array of coefficients of the fitted polynomial returned by POLFIT. P(i) = coefficient of X^{i-1}.

N integer input variable, the number of data points.

MA (and M) input - for fitting to a single polynomial: MA is an integer specifying the order of the polynomial. P must have dimension at least M=MA+1.

input - for fitting to a sequence of polynomials: MA is an integer array, dimensioned MA(4) with
MA(1) = 0
MA(2) = minimum order of polynomial
MA(3) = maximum order
MA(4) = increment in order

In this case, P must have dimension at least M=MA(3)+1.

E(M,M) real working storage array for internal use by POLFIT.

NE integer, number of rows dimensioned for E in the calling program.

S — real sum of squares of residuals returned from POLFIT

$$s = \sum_{i=1}^{n} w(i)(y(i)-yf(i))^2$$

WS(NWS) — real working storage array for internal use by POLFIT. Must be dimensioned with NWS $\geq$ m(4+n+2m) in the calling program.

ROUTINES CALLED — PRESET from AELIB

STORAGE REQUIRED — 1300_8

EXIT — For fitting to a single polynomial: No messages are printed and fitted vaues, coefficients, and sum of squares are returned to the calling program.

For fitting to a sequence of polynomials: the coefficients are printed after each fit under the heading:

PARAMETERS FOR FIT OF ORDER XXX ARE

The fitted values, coefficients, and sum of squares returned to the calling program are those for highest order fit.

(Note: A fit of order n achieved via a multiple fit of maximum order >n will not be identical to a single fit of order n because the orthogonality of legendre polynomials used is only approximate.)

SPEED — Depends on N, MA, and whether single or multiple fits are being performed.

The results of three timing tests are:

1) Single fitting, N=33, MA=5 required approximately .02 s (6600, Sept. 1974).
2) Single fitting, N=33, MA=2 required approximately .008 s (6600, Sept. 1974).
3) Multiple fitting, N=33, MA=2,3,4,5,6 required approximately .045 s including printing of results. (6600, Sept. 1974)

EXAMPLES

The following program fits a polynomial of the form $y = a + bx + cx^2 + dx^3$ to a maximum of 25 data points (x,y) read in from cards.

1. Single Fit of Order 3

```
      PROGRAM USE(INPUT,OUTPUT,TAPE5=INPUT)
      REAL X(25),Y(25),W(25),YF(25),P(4),E(4,4),WS(148)
      MA=3
      NE=4
C
C     READ THE DATA POINTS
C
      N=1
   20 READ(5,1000)X(N),Y(N),W(N)
 1000 FORMAT(3E12.6)
      IF(EOF(5))15,10
   10 N=N+1
      IF(N.LE.25) GO TO 20
   15 N=N-1
C
C     DO FIT
C
      CALL POLFIT(X,Y,W,YF,P,N,MA,E,NE,S,WS)
C
C     PRINT RESULTS
C
      PRINT 2000
 2000 FORMAT(1H1,4X,4HX(N),16X,4HY(N),16X,4HW(N),16X,5HYF(N))
      PRINT 3000,(X(I),Y(I),W(I),YF(I),I=1,N)
 3000 FORMAT(4(1X,E12.6,7X))
      PRINT 4000,(P(I),I=1,NE)
 4000 FORMAT(1X,E12.6)
      PRINT 5000,S
 5000 FORMAT(1H0,30HSUM OF SQUARES OF RESIDUALS IS,E12.6)
      STOP
      END
```

2. <u>To Convert the Above Program to do Multiple Fit of Orders 1, 2 and 3</u>

Replace card

MA=3

with

```
INTEGER MA(4)
MA(1)=0
MA(2)=1
MA(3)=3
MA(4)=1
```

```
      PROGRAM USE(INPUT,OUTPUT,TAPE5=INPUT)
      REAL X(25),Y(25),W(25),YF(25),P(4),E(4,4),WS(148)
      INTEGER MA(4)
      MA(1)=0
      MA(2)=1
      MA(3)=3
      MA(4)=1
      NE=4
C
C     READ THE DATA POINTS
C
      N=1
   20 READ(5,1000)X(N),Y(N),W(N)
 1000 FORMAT(3E12.6)
      IF(EOF(5))15,10
   10 N=N+1
      IF(N.LE.25) GO TO 20
   15 N=N-1
C
C     DO FIT
C
      CALL POLFIT(X,Y,W,YF,P,N,MA,E,NE,S,WS)
C
C     PRINT RESULTS
C
      PRINT 2000
 2000 FORMAT(1H1,4X,4HX(N),16X,4HY(N),16X,4HW(N),16X,5HYF(N))
      PRINT 3000,(X(I),Y(I),W(I),YF(I),I=1,N)
 3000 FORMAT(4(1X,E12.6,7X))
      PRINT 4000,(P(I),I=1,NE)
 4000 FORMAT(1X,E12.6)
      PRINT 5000,S
 5000 FORMAT(1H0,30HSUM OF SQUARES OF RESIDUALS IS,E12.6)
      STOP
      END
```

POLFIT

AUTHOR	L.E. Evans	DATE	September 1974
REVISED	M.T. Boulanger & C.A. Wills		April 1980
	C.A. Wills		May 1981

TITLE Multiple Linear Regression to a Polynomial of Specified Degree in One or More Independent Variables

ENTRY Call PRFIT(IDEG,X,NRX,Y,W,YFIT,M,N,B,KTRM,NRK,IH,NT,IER,WS)

IDEG Integer input variable, the highest degree of polynomial to be fitted. $1 \le$ IDEG.

X(NRX,M) Real input matrix of data points such that x(i,j) is the ith data value of the jth independent variable. NRX $\ge$ N.

NRX Integer input variable, the number of rows dimensioned for X in the calling program.

Y(N) Real input array of data values of the dependent variable.

W(N) Real input array of weights for the dependent variable. For a weighted fit use w(i) = $(\text{variance}(y(i)))^{-1}$. For an unweighted fit, use w(i) = 1.

YFIT(N) Real output array of fitted values of the dependent variable returned from PRFIT.

M Integer input variable, the number of independent variables.

N Integer input variable, the number of data points.

B(NT) Real output array of coefficients of the fitted polynomial returned by PRFIT. b(i) is coefficient of the (i-1)th non-constant term and b(1) is the constant term.

KTRM(NRK,NCK) Integer matrix returned from PRFIT defining the non-constant terms in order of the polynomial being fitted. NRK $\ge$ M, NCK $\ge$ NT-1. KTRM(i,j) is the power of the ith independent variable in the jth non-constant term of the regression polynomial.

NRK — Integer input variable, the number of rows dimensioned for KTRM in the calling program.

IH(NT) — Integer array returned from PRFIT as a vector of plus and minus ones. IH(i)=-1 implies that the ith term is not in the fitted polynomial. IH(i)=1 implies that the ith term is in the polynomial.

NT — Integer input variable, to specify the number of terms in the polynomial to be fitted. The value for NT must be $\frac{(M+IDEG)!}{M!IDEG!}$

IER — Integer, error flag returned from PRFIT. If no errors have occurred in PRFIT, IER=0.

IER=37 or 129,130,131, or 132 indicate that an error was detected in RLSTP. In particular, IER=130 means that RLSTP was undecided on the best possible fit. (For more information on these error conditions, see the IMSL write-up for RLSTP.)

WS(NRWS) — Real working storage array for internal use by PRFIT. NRWS $\geq$ (NT*(2*N+NT+3))/2 + 1 + M. The first (NT)*(NT+1)/2 words in this array are used to pass the sums of squares and cross products matrix to RLSTP and on return from PRFIT will contain the error information set up by RLSTP and described in the IMSL write-up. The AELIB routine PSTATF may be called after PRFIT to further analyse this error information.

ROUTINES CALLED — COMKTM, POLREG from AELIB (POLREG calls SSCP, PRESET and PCALC from AELIB and RLSTP from IMSLIB)

EXITS — Normal Exit: If no errors fatal to the execution of PRFIT have been detected, then control is returned to the calling routine with the fitted values in YFIT, the fitted polynomial defined in KTRM, IH and B and the error information from RLSTP in WS. IER may be one of 0 (no errors) or either 37 or 130 (non-fatal errors in RLSTP) in this case.

Error Exit: Errors fatal to the execution of PRFIT cause control to be returned to the calling routine with elements of arrays YFIT, B and WS used by PRFIT set to indefinite and elements of the array IH used by PRFIT set to 0. IER values of 129, 131, or 132 represent errors considered fatal to PRFIT. Suitable error messages are printed from RLSTP.

SPEED

Depends mainly on the number of terms (NT) in the polynomial and the number of data points (N) and to a lesser extent on the number of independent variables (M) in the polynomial.

The time the program example takes to execute the PRFIT routine is approximately 0.03 seconds. (6600, Sept. 1975)

PROGRAM EXAMPLE

The following program fits a polynomial of order 3 in 2 independent variables to 20 data points read in from cards.

If fatal error conditions have been detected, the program is terminated. Otherwise, output from this program is as follows:

1) The data values for the independent and dependent variable, the fitted values, and the absolute and percentage errors in the dependent variable are printed.

2) The fit is then described by tabulating the KTRM matrix defining the terms in the polynomial, the IH array specifying the significant terms in the polynomial, and the B array of coefficients of those terms.

3) A plot is made of the original data, the fitted polynomial, and the 95% confidence interval for the fitted polynomial.

```
      PROGRAM TEST (INPUT,OUTPUT,TAPE5=INPUT,PLOT)
      REAL X(25),Y(25), W(25), YFIT(25)
      REAL B(2), WS(60), WSPLOT(18)
      INTEGER IH(2), KTRM(1,1)
      COMMON/FORMAT/XL,YL,FTITLE(8),FSUBTIT(8),FXAXTIT(8),FYAXTIT(8),
     ,NSTVARS,STVARS(10)
C
C
      DATA W/25 * 1.0/
      IDEG = 1
      NRX = 25
      M = 1
      N = 25
      NRK = 1
      NT = 2
C
```

```
C     READ IN DATA
C
      DO 10 I=1,N
           READ (5,*) X(I),Y(I)
   10 CONTINUE
C
C     DO FIT AND TEST ERROR FLAG
C
      CALL PRFIT(IDEG,X,NRX,Y,W,YFIT,M,N,B,KTRM,NRK,IH,NT,IER,WS)
      IF(IER .GT. 37 .AND. IER .NE. 130) GO TO 995
C
C     PRINT RESULTS
C
      DO 20 I=1,N
         ERROR = Y(I) - YFIT(I)
         PRCNTER = ABS(ERROR) / Y(I) * 100.0
         PRINT 1000, X(I), Y(I), YFIT(I), ERROR, PRCNTER
   20 CONTINUE
 1000 FORMAT(1H ,5(F8.3,4X))
C
C     PRINT DESCRIPTION OF POLYNOMIAL
C
      PRINT 2000, 1H0, IH(1), B(1)
 2000 FORMAT(1H-,24X,A1,I20,G20.7)
      DO 30 I=2,NT
         PRINT 3000, KTRM(1,I-1), IH(I), B(I)
   30 CONTINUE
 3000 FORMAT(1H , 5X, 2I20, G20.7)
C
C     SET UP TITLES FOR PLOT
C
      PROB = 0.05
      LOPT = 1
      FTITLE(1) = 10H(*CIPLOT E
      FTITLE(2) = 10HXAMPLE*)
      FXAXTIT(1) = 10H(*X-AXIS T
      FXAXTIT(2) = 10HITLE*)
      FYAXTIT(1) = 10H(*Y-AXIS T
      FYAXTIT(2) = 10HITLE*)
C
C     CALCULATE AND PLOT CONFIDENCE INTERVAL ON PREDICTED VALUES
C
      CALL CIPLOT(X,Y,W,YFIT,N,B,KTRM,IH,NT,WS,PROB,WSPLOT,LOPT)
      GO TO 999
C
C     TERMINATE PROGRAM DUE TO FATAL ERROR
C
  995 PRINT 996
  996 FORMAT(7H IER = ,I5,33HINDICATES FATAL ERROR(FROM RLSTP))
  999 STOP
      END
```

AUTHORS	D.E. Smith and L.E. Evans	Date	June 1975
REVISED	C.A. Wills		May 1981
	C.A. Wills		March 1982

TITLE Multiple linear regression to an arbitrary polynomial in one or more independent variables.

ENTRY CALL POLREG(X,NRX,Y,W,YFIT,M,N,B,KTRM,NRK,IH,NT,IER,WS)

X(NRX,M) Real input matrix of data points such that x(i,j) is the ith data value of the jth independent variable. NRX ≥ N.

NRX Integer input variable, the number of rows dimensioned for X in the calling program.

Y(N) Real input array of data values of the dependent variable.

W(N) Real input array of weights for the dependent variable. For a weighted fit, use w(i)=(variance(y(i)))$^{-1}$. For an unweighted fit, use w(i)=1.

YFIT(N) Real output array of fitted values of the dependent variable returned from POLREG.

M Integer input variable, the number of independent variables.

N Integer input variable, the number of data points.

B(NT) Real output array of coefficients of the fitted polynomial returned by POLREG. b(i) is the coefficient of the (i-1)st non-constant term and b(1) is the constant term.

KTRM(NRK,NCK) Integer input matrix defining the non-constant terms, in order, of the polynomial to be fitted. NRK ≥ M, NCK ≥ NT-1. KTRM(i,j) is the power of the ith independent variable in the jth term of the regression polynomial.

NRK Integer input variable, the number of rows dimensioned for KTRM in the calling program.

IH(NT) Integer array returned from POLREG as a vector of plus and minus ones. IH(i)=-1 implies that the ith term is not in the fitted polynomial. IH(i)=1 implies that the ith term is in the polynomial.

NT Integer input variable, the number of terms in the polynomial to be fitted.

IER Integer, error flag returned from POLREG. If no errors have occurred in POLREG, IER=0.

IER=37 or 129,130,131 or 132 indicate that an error was detected in RLSTP. In particular, IER=130 means that RLSTP was undecided on the best possible fit. (For more information on these error conditions, see the IMSL write-up for RLSTP.)

WS(NRWS) Real working storage array for internal use by POLREG. NRWS ≥ (NT*(2*N+NT+3))/2 + 1 + M. The first (NT)*(NT+1)/2 words in this array are used to pass the sums of squares and cross products matrix to RLSTP and on return from POLREG will contain the error information set up by RLSTP and described in the IMSL write-up. The routine PSTAT may be called after POLREG to further analyse this information.

ROUTINES CALLED SSCP, PRESET and PCALC from AELIB. RLSTP from IMSLIB.

EXIT Normal Exit: If no errors fatal to the execution of POLREG are detected, then control is returned to the calling routine with the fitted values in YFIT, the fitted polynomial defined in KTRM, IH and B and the error information from RLSTP in WS. IER will be one of 0 (no errors) or either 37 or 130 (non-fatal errors in RLSTP) in this case.

Error Exit: If errors fatal to the execution of POLREG are detected, then control is returned to the calling routine with elements of arrays YFIT, B and WS used by POLREG set to indefinite and the elements of the array IH used by POLREG set to 0. IER values of 129, 131, and 132 represent errors considered fatal to PRFIT. Suitable error messages are printed from RLSTP.

SPEED Depends mainly on the number of terms (NT) in the polynomial and the number of data points (N) and to a lesser extent on the number of independent variables (M) in the polynomial.

The time the program example takes to execute the POLREG routines is approximately 50 ms. (6600, Sept. 1975)

EXAMPLE The following program fits a user defined polynomial in two independent variables to 20 data points read in from cards.

For this example, the polynomial to be fitted is:

$$P(x_1,x_2) = b_1 + b_2x_1x_2 + b_3x_1x_2^2 + b_4x_1^3x_2$$

$$+ b_5x^3x_2^2 + b_6x_1^2x_2^3 + b_7x_1^4x_2^2 + b_8x_2^8$$

and the KTRM matrix for this polynomial is:

$$\text{KTRM} = \begin{pmatrix} 1 & 1 & 3 & 3 & 2 & 4 & 0 \\ 1 & 2 & 1 & 2 & 3 & 2 & 8 \end{pmatrix}$$

Note that the KTRM matrix does not include the constant term.

If fatal errors are detected, the program is terminated. Otherwise, output from this program is as follows:

1) The data values for the independent and dependent variable, the fitted values, absolute error and percentage error are printed.

2) The resulting fit is then described by tabulating the KTRM matrix defining the terms in the polynomial, the IH array specifying the significant terms in the polynomial, and the B array of coefficients of those terms.

```
      PROGRAM USE(INPUT,OUTPUT)
      REAL B(8),YFIT(50),WS(300)
      REAL X(50,2),Y(50),W(50)
      INTEGER KTRM(2,7),IH(8)
      DATA W/50*1.0/
      DATA KTRM/1,1,1,2,3,1,3,2,2,3,4,2,0,8/
      M=2
      N=20
      NT=8
      NRX=50
      NRK=2
C
C     READ IN DATA
C
      DO 20  I = 1, N
        READ 1000, (X(I, J), J = 1, 2), Y(I)
   20 CONTINUE
C
C     DO FIT AND TEST ERROR FLAG
C
      CALL POLREG(X,NRX,Y,W,YFIT,M,N,B,KTRM,NRK,IH,NT,IER,WS)
      IF(IER.GT.37 .A. IER.NE.130) GO TO 900
C
C     PRINT RESULTS
C
C     PRINT DATA AND FITTED VALUES
C
      DO 40  I = 1, N
        ERROR = Y(I) - YFIT(I)
        PRCNTER = ABS(ERROR) / Y(I) * 100.0
        PRINT 1001, (X(I, J), J = 1, 2), Y(I), YFIT(I), ERROR, PRCNTER
   40 CONTINUE
C
C     PRINT DESCRIPTION OF POLYNOMIAL
C
      PRINT 1002, 1H0, 1H0, IH(1), B(1)
C
      DO 50  I=2,NT
         PRINT 1003,KTRM(1,I-1),KTRM(2,I-1),IH(I),B(I)
   50 CONTINUE
      GO TO 999
C
C     TERMINATE PROGRAM DUE TO FATAL ERRORS
  900 PRINT 1004
  999 STOP
1000  FORMAT(1H , 3F8.4)
1001  FORMAT(1H , 10(F8.3, 4X))
1002  FORMAT(1H-, 24X, A1, 1(19X, A1), I20, G20.7)
1003  FORMAT(1H , 5X, 3I20, G20.7)
 1004 FORMAT(1H ,"IER FROM POLREG INDICATES FATAL ERRORS")
      END
```

AUTHORS D.E.' Smith and L.E. Evans Date June 1975

REVISED C.A. Wills May 1981

TITLE

Define a polynomial of given degree in one or more independent variables in form suitable for use by multiple linear regression routines

ENTRY

CALL COMKTM(IDEG,KTRM,NRK,M)

IDEG — Integer input variable, the highest degree of polynomial to be fitted. $1 \leq \text{IDEG}$.

KTRM(NRK,NCK) — Integer matrix returned from COMKTM defining the non-constant terms in order of the polynomial being fitted.

$\text{NRK} \geq \text{M}$, $\text{NCK} \geq \text{NT}-1$, where $\text{NT} = \dfrac{(m+\text{IDEG})!}{m!\,\text{IDEG}!}$

is the number of terms in the polynomial. KTRM(i,j) is the power of the ith independent variable in the jth term of the regression polynomial.

NRK — Integer input variable, the number of rows dimensioned for KTRM in the calling program.

M — Integer input variable, the number of independent variables.

ROUTINES CALLED

PRESET from AELIB.

EXIT

Control is returned to the calling program with the terms of the polynomial defined in KTRM.

SPEED

Depends mainly on the highest degree (IDEG) of the polynomial and the number of independent variables (M) in the polynomial.

The time the example program takes to execute COMKTM is approximately .25 ms. (6600, Sept. 1975)

EXAMPLE

See writeup for SSCP. Combined use of the routines COMKTM, SSCP and PCALC is illustrated as an alternative to PRFIT or POLREG.

AUTHORS	D.E. Smith and L.E. Evans	DATE	August 1975
REVISED	B.E. Purcell	DATE	March 1978

TITLE Weighted sum of squares and cross products calculation for use with IMSL routine RLSTP to do multiple linear regression to a polynomial.

ENTRY CALL SSCP(X,NRX,Y,W,M,N,KTRM,NRK,NT,WS)

X(NRX,M) Real input matrix of data points such that x(i,j) is the ith data value of the jth independent variable. NRX $\geq$ N.

NRX Integer input variable, the number of rows dimensioned for X in the calling program.

Y(N) Real input array of data values of the dependent variable.

W(N) Real input array of weights for the dependent variable. For a weighted fit, use $w(i)=(\text{variance}(y_i))^{-1}$. For an unweighted fit, use w(i)=1.

M Integer input variable, the number of independent variables.

N Integer input variable, the number of data points.

KTRM(NRK,NCK) Integer input matrix defining the non-constant terms in order, of the polynomial being fitted. NRK $\geq$ M, NCK $\geq$ NT-1. KTRM(i,j) is the power of the ith independent variable in the jth non-constant term of the regression polynomial.

NRK Integer input variable, the number of rows dimensioned for KTRM in the calling program.

NT Integer input variable, the number of terms in the polynomial to be fitted.

WS(NRWS) Real working storage area. On return from SSCP the corrected sums of squares and cross products matrix will be stored in

WS(1) to WS((NT*(NT+1))/2. NRWS $\geq$ (NT*(2*N+NT+3))/2 + 1.

EXITS

Control is returned to the calling program with the corrected sums of squares and cross products matrix stored in WS ready for a subsequent call to RLSTP.

SPEED

Depends mainly on the number of terms (NT) in the polynomial and the number of data points (N) and to a lesser extent on the number of independent variables (M) in the polynomial.

The time the program example takes to execute SSCP is approximately 9.7 ms. (6600, Sept. 1975)

EXAMPLE

The following program illustrates the use of the routines COMKTM, SSCP, PCALC and PSTATF in performing a weighted fit using RLSTP directly. The program is similar to that provided for PRFIT but allows the significance levels, ALPHAI and ALPHAO for RLSTP to be variable.

A polynomial of order 3 in two independent variables is fitted to 20 data points read in from cards. The description of this polynomial is set up by COMKTM, the corrected sums of squares and cross products matrix calculated by SSCP, the fit done by RLSTP and the constant term and fitted values provided by PCALC. A subsequent call to PSTATF calculates the residual sum of squares for the fit, an error matrix for the fitted coefficients, and standard deviation estimates for the fitted function values.

If fatal error conditions are detected, the program is terminated. Otherwise, output is as follows:

1) The data values for the independent and dependent variable, the fitted values, standard deviation estimates, and the absolute and percentage errors in the dependent variable are printed.

2) The fit is then described by tabulating the KTRM matrix defining the terms in the polynomial, the IH array specifying the significant terms in the polynomial, and the B array of coefficients of those terms.

3) the residual sum of squares and parameter error matrix are then printed.

```
      PROGRAM USE3(INPUT,OUTPUT)
      REAL X(50,2),Y(50),W(50)
      REAL B(10),YFIT(50),WS(300)
      DIMENSION SDY(50)
      INTEGER KTRM(2,9),IH(10),JX(9)
      REAL E(10,10)
      DATA W/50*1.0/
      M = 2
      N=20
      NT=10
      NRX=50
      NRK = 2
      IDEG=3
C
C     READ IN DATA
C
      DO 20  I = 1, N
        READ 1000, (X(I, J), J = 1, 2), Y(I)
   20 CONTINUE
C
C     SET UP KTRM MATRIX DESCRIBING POLYNOMIAL
C
      CALL COMKTM(IDEG,KTRM,NRK,M)
C
C     CALCULATE CORRECTED SUMS OF SQUARES AND CROSS PRODUCTS MATRIX
C     FOR INPUT TO RLSTP.
C
C
      CALL SSCP(X,NRX,Y,W,M,N,KTRM,NRK,NT,WS)
C
C     DO FIT USING RLSTP AND TERMINATE ON ERROR CONDITIONS.
C
      IOPT=0
      IH(1)=0
      NTM1=NT-1
      DO 30 I=1,NTM1
           JX(I)=0
   30 CONTINUE
      ALFAI=ALFAO=0.10
      CALL RLSTP(WS,NTM1,N,ALFAI,ALFAO,JX,IH,B,IOPT,IER)
C
C     IF(IER.GT.37 .A. IER.NE.130) GO TO 900
C
C     COMPUTE THE CONSTANT TERM AND FITTED VALUES
C
      CALL PCALC(YFIT,N,B,IH,NT,WS)
C
C     COMPUTE  ERROR STATISTICS
C
      NRE=10
      ICON=1
      CALL PSTATF(ICON.IH.NT.N.E.NRE.SDY.SS.WS)
```

```
C
C     PRINT RESULTS
C
      DO 40  I = 1, N
        ERROR = Y(I) - YFIT(I)
        PRCNTER = ABS(ERROR) / Y(I) * 100.0
        PRINT 1001, (X(I, J), J = 1, 2), Y(I), YFIT(I),SDY(I), ERROR, PR
     ,CNTER
   40 CONTINUE
C
C     PRINT DESCRIPTION OF POLYNOMIAL
C
      PRINT 1002, 1H0, 1H0, IH(1), B(1)
      DO 50 I=2,NT
         PRINT 1003,KTRM(1,I-1),KTRM(2,I-1),IH(I),B(I)
   50 CONTINUE

      PRINT 1004,SS
      PRINT 1005,((E(I,J),J=1,NT),I=1,NT)
  900 STOP
 1000 FORMAT(1H , 3F8.4)
 1001 FORMAT(1H , 10(F8.3, 4X))
 1002 FORMAT(1H-, 24X, A1, 1(19X, A1), I20, G20.7)
 1003 FORMAT(1H , 5X, 3I20, G20.7)
```

AUTHORS	D.E. Smith and L.E. Evans	DATE	August 1975
REVISED	C.A. Wills		May 1981

TITLE Compute the constant term to complete the fit provided by RLSTP and calculate from this set of coefficients the fitted values of the dependent variable.

ENTRY CALL PCALC(YFIT,N,B,IH,NT,WS)

Note: This routine is to be used only after successful calls to SSCP and RLSTP. The parameters, N,B,IH,NT and WS, must remain unchanged between the consecutive calls of SSCP, RLSTP and PCALC.

YFIT(N) Real output array of fitted values of the dependent variable returned by PCALC.

N Integer input variable, the number of data points.

B(NT) Real array of coefficients of the fitted polynomial returned by RLSTP and required as input for PCALC. On return from PCALC, b(i) is the coefficient of the (i-1)st non-constant term and b(1) is the constant term.

IH(NT) Integer array returned by RLSTP as a vector of plus and minus ones and is required as input to PCALC. On return from PCALC, IH(i)=-1 implies that the ith term (including constant as first term) is not in the fitted polynomial. IH(i)=1 implies that the ith term is in the polynomial.

NT Integer input variable, the number of terms in the polynomial supplied to PCALC.

WS(NRWS) Real working storage array. The first (NT*(NT+1))/2 words in this array contain the error information set up by the RLSTP routine described in the IMSL writeup and are not used by PCALC. The remaining storage from index (NT*(NT+1))/2 to (NT*(2*N+NT+3))/2 must be the values as they were set up to SSCP. NRWS ≥ (NT*(2*N+NT+3))/2 + 1.

EXIT Control is returned to the calling program with the fitted values in YFIT, the absence or presence of terms of the polynomial in IH, with the coefficients in B.

SPEED Depends mainly on the number of terms (NT) in the polynomial and the number of data points (N) and to a lesser extent on the number of independent variables (M) in the polynomial.

The time the example program takes to execute the PCALC routine is approximately 1.1 ms. (6600, Sept. 1975)

EXAMPLE See writeup for SSCP. Combined use of the routines COMKTM, SSCP and PCALC is illustrated as an alternative to PRFIT or POLREG.

AUTHORS D.E. Smith and L.E. Evans DATE: August 1975

REVISED C.A. Wills May 1981

TITLE

Error statistics calculation for the polynomial fit from RLSTP providing the covariance and correlation of pairs of coefficients, the standard deviation estimates for the coefficients and fitted function values, and the (weighted) sum of squares of residuals.

ENTRY

CALL PSTATF(ICON,IH,NT,N,E,NRE,SDY,SS,WS)

(This routine may be called after AELIB routines PRFIT, POLREG, or RLSTP. If called after RLSTP but before PCALC, the constant term is not part of the fit and statistics for it on the fitted function values are, therefore, not available. If called after PRFIT, POLREG or PCALC following RLSTP, the constant term is included in the fit and statistics for it and the fitted function values are calculated.)

ICON Integer input variable, which should be 0 if the constant term is not included in the fit and 1 if the constant term is included. ICON should be set to 0 only if PSTATF is called directly after a call to RLSTP. Otherwise ICON should be 1.

IH(NT) Integer input array set up by a prior call to one of the four routines mentioned above as a vector of plus or minus ones. IH(i)=-1 implies the ith term is not in the fitted polynomial. IH(i)=1 implies that the ith term is in the polynomial.

NT Integer input variable, the number of terms in the polynomial supplied to PCALC, PRFIT or POLREG (including the constant term).

N Integer input variable, the number of data points.

E(NRE,NT) Real array, NRE $\geq$ NT, returned by PSTATF containing the error matrix defined as follows:

E(i,j)=0 if either the ith or jth term are not in the fitted model.

E(i,j)= standard deviation of term i if i=j (diagonal)
covariance of ith and jth terms if i<j (above diagonal)
correlation of ith and jth terms if i>j (below diagonal)

NRE Integer input variable, the number of rows dimensioned for E in the calling program.

SDY(N) Real output array. If ICON=1, SDY returns the standard deviation estimates for the fitted function values. If ICON=0, this array is set to indefinites.

SS Real returned by PSTATF as the (weighted) sum of squares of residuals.

WS(NRWS) Real working storage array containing data set up by a prior call to PRFIT, POLREG or RLSTP.

The first (NT*(NT+1))/2 words contain the error information set up by RLSTP and will be altered by PSTATF. Also required as set up by PRFIT, POLREG or SSCP are the means of the non-constant terms of the polynomial sorted from index (NT*(NT+1))/2 + 1 to (NT*(NT+1))/2 + NT-1 the mean of the dependent variable stored at index (NT*(NT+1))/2 + NT, and the sum of weights stored at index (NT*(2*N+NT+3))/2 + 1).
NRWS ≥ (NT*(2*N+NT+3))/2 + 1

ROUTINES CALLED RLSUBM from IMSLIB.

EXIT Control is returned to the calling program with the statistics stored in the matrices E and SDY, and the (weighted) sum of squares of residuals in SS.

SPEED Depends primarily on the number of terms (NT) in the polynomial and also on whether or not the constant term is included.

The time required for the execution of PSTATF in the program example is approximately 1.2 ms. (6600, Sept. 1975)

EXAMPLE	See writeup for SSCP. The use of PSTATF is illustrated along with COMKTRM, SSCP, RLSTP and PCALC.		
AUTHORS	D.E. Smith and L.E. Evans	DATE	August 1975
REVISED	L.E. Evans	DATE	March 1977
	C.A. Wills		May 1981

TITLE For a regression equation in one independent variable found by RLSTP, POLREG, or PRFIT plot a complete frame with axes and labels containing the original data, the fitted curve, and either the confidence limits for the predicted mean values or for the predicted individual values of the dependent variable.

INTRODUCTION

The model fitted by RLSTP, POLREG, or PRFIT has an associated variance as expressed in the error matrix for the fitted parameters. CIPLOT uses this information to calculate confidence intervals either for predicted mean values (1) or for predicted individual values of the dependent variable (2). If, for a given value of the independent variable x_i the predicted value of the dependent variable is y_i, then the two kinds of confidence intervals for y_i are:

(1) $y_i \pm t(n\text{-}nt,\ p/2)\ \sigma_i$

(2) $y_i \pm t(n\text{-}nt,\ p/2)\ \sqrt{\sigma_i^2 + ss}$ (unweighted)

$y_i \pm t(n\text{-}nt,\ p/2)\ \sqrt{\sigma_i^2 + 1.0/w(i)}$ (weighted)

where w, n, nt, ss are as defined for PRFIT or PSTATF, σ_i is the standard deviation estimate for the predicted value y_i, t is Student's t distribution, and p defines the confidence interval.

ENTRY CALL CIPLOT(X,Y,W,YFIT,N,B,KTRM,IH,NT,WS,PROB,WSPLOT,LOPT,IER)

Note: This routine is to be used only after successful calls to RLSTP, POLREG, or PRFIT. The parameters X, Y, W, YFIT, B, KTRM, IH, NT, and WS remain unchanged from those calls. If RLSTP has been called directly the routines KTRM, COMKTM, SSCP, and PCALC must also have been called before calling CIPLOT.

The PLOT file must be defined on the PROGRAM card. Plots are made from CIPLOT via calls to SIMPLT and PLODA, AELIB routines number 2-1-20 and 2-1-03. The plot symbol is defined to be .08" in size and the length of the dash cycle (D1, B1, D2, B2) is defined to be (0.4", 0.2", 0.2", 0.2"). Non-default parameters for the plot may be supplied in COMMON blocks as described in the SIMPLT writeup with the one exception that the subtitle FSUBTIT should not be changed by the user.

X(N) Real input matrix of data points of the independent variable

Y(N) Real input array of data values of the dependent variable

W(N) Real input array of weights for the dependent variable. For a weighted fit, use w(i)=1.0/variance(y(i)). For an unweighted fit, use w(i)=1.

YFIT(N) Real input array set up by a prior call to PCALC, PRFIT, or POLREG containing the fitted values of the dependent variable.

N Integer input variable, the number of data points.

B(NT) Real input array of coefficients of the fitted polynomial returned by PCALC, POLREG, or PRFIT. B(i) is the coefficient of the (i-1)st non-constant term and B(1) is the constant term.

KTRM(1,NCK) Integer input matrix defining the non-constant terms, in order of the polynomial fitted by RLSTP, PRFIT, or POLREG. NCK ≥ NT-1.

IH(NT) Integer input array containing a vector of plus or minus ones as defined for PRFIT or POLREG. IH(i)=-1 implies the ith term is not in the fitted polynomial. IH(i)=1 implies that the ith term is in the polynomial.

NT Integer input variable, the number of terms including the constant in the polynomial supplied to PCALC, PRFIT, or POLREG. To be able to calculate confidence intervals N must be greater than NT.

WS(NRWS) Real working storage array containing data set up by a prior call to PRFIT, POLREG, or RLSTP. The first (NT*(NT+1))/2 words contain the error information, the means of the non-constant terms of the polynomial are stored from index (NT*(NT+1))/2 + 1 to (NT*(NT+1))/2 + NT, and the sum of weights stored at index (NT*(2*N+NT+3))/2 + 1. NRW ≥ (NT*(2*N+NT+3))/2 + 1 + M.

PROB Real input value, 100*(1-PROB)% confidence limits will be calculated.

WSPLOT(NRW2) Real working storage array for internal use by CIPLOT. (NRW2 $\geq$ max(NT*NT, 18))

LOPT Integer input variable set to 1 for confidence interval for mean values or to 2 for confidence interval for predicted values.

ROUTINES CALLED PLODA, PSTATF, SIMPLT from AELIB, MDSTI, VMULFS from IMSLIB. SUBPLOT a utility routine loaded with CIPLOT.

EXIT If no fatal errors have occurred CIPLOT will terminate by a normal exit which returns control to the calling routine.

If an error occurs control is returned to the calling routine but no plot will be generated. Possible values of IER and associated errors are described below:

IER	Description
0	No errors.
1	Fatal error. The value of PROB is not between 0 and 1. The message ***PROB DOESN'T LIE BETWEEN ZERO AND ONE-NO PLOT PRODUCED is printed.
2	Fatal error. The value of N is less than or equal to the value of NT. The message ***DEGREES OF FREEDOM (N-NT) ARE LESS THAN OR EQUAL TO ONE-NO PLOT PRODUCED is printed.
3	Fatal error. An internal error has occurred in the IMSL subroutine MDSTI. The message ***INTERNAL ERROR IN IMSL ROUTINE MDSTI-NO PLOT PRODUCED is printed.

SPEED This depends on the number of terms in the polynomial NT and the number of data points N. The execution time for CIPLOT in the program example is 0.15 seconds (CYBER 170 Model 175, November 1981).

EXAMPLE See writeup for PRFIT. The use of CIPLOT after PRFIT is illustrated.

AUTHORS C.A. Wills and M.T. Boulanger DATE 1981 November

TITLE Weighted Least Squares Fitting of Data to a Function of One or More Independent Variables With or Without User Supplied Partial Derivatives.

ENTRY CALL MLSQQ(IOPT,X,NRX,Y,W,YF,P,NC,N,M,AUX,E,NE,S,RERR, WS,IER)

IOPT integer input variable to define the type of solution required as follows:

IOPT	Derivatives supplied in AUX†	Solution Method for Normal System
0	No	Marquardt*
1	Yes	Marquardt*
-1	Yes	Newton-Raphson**
-2	No	Newton-Raphson**

† if derivatives are not supplied, they are approximated by MLSQQ.

* should be chosen if fitting function is non-linear in parameters.

** must be chosen if fitting function is linear.

X(NRX,NCX)

real input array of values of the independent variables, X(i,j) being the ith data value of the jth independent variable.

NRX ≥ N, NCX = number of independent variables.

NRX number of rows of X as dimensioned in the calling program.

Y(N) real input array of values of the dependent variable.

W(N) real input array of weights for dependent variable. For a weighted fit, estimates of the variances of the dependent variable values should be used as follows:

W(i) = 1/var(Y(i)), i=1,...,N

For an unweighted fit, any non-zero constant value may be used but

W(i) = 1, i=1,...,N

is recommended.

YF(N) real output array of fitted values of dependent variable returned from MLSQQ.

P(M) real input and output array of parameters for user supplied function. Initial values are passed to MLSQQ and final values returned from MLSQQ.

NC(M) integer input array specifying whether parameters are to be fixed or varied. NC(i)=0 means ith parameter fixed and NC(i)=1 means ith parameter varied. MU is the number of non zero entries in NC which defines the number of parameters to be varied.

N integer, number of data points.

M integer, number of parameters in the fitting function.

AUX subroutine supplied by the user to calculate the fitting function. It must be declared EXTERNAL in the main program. Its calling sequence and arguments are described below.

E(NE,M) real array, NE $\geq$ N, used in two different ways by MLSQQ:

(1) E(N,M) is used internally by MLSQQ to store the negative of the Jacobian of the user's function at each iteration.

(2) E(M,M) is an output array containing the estimated error matrix for the fitted parameters as follows:

E(i,j)=0 if P(i) and/or P(j) are fixed parameters; otherwise,

$$E(i,j) = \begin{cases} \sqrt{\mathrm{var}(P(i))}, & i=j, i=1,\ldots,m \text{ (diagonal)} \\ \mathrm{cov}(P(i),P(j)), & j=2,\ldots,m; \text{ (above} \\ & i=1,\ldots,j-1 \text{ diagonal)} \\ \dfrac{\mathrm{cov}(P(i),P(j))}{\sqrt{\mathrm{var}(P(i)).\mathrm{var}(P(j))}}, & i=2,\ldots,m; \text{ (below} \\ & j=1,\ldots,i-1 \text{ diagonal)} \end{cases}$$

NE integer, number of rows of E as dimensioned in calling program.

S real output variable to return the weighted residual sum of squares from MLSQQ.

$$S = \sum_{i=1}^{N} W(i)(Y(i)-YF(i))^2$$

RERR real input variable specifying relative error necessary for convergence of MLSQQ. If the value of the ith parameter after jth iteration is $P_{i,j}$, then MLSQQ has converged if

$$\left|\frac{P_{i,j}-P_{i,j-1}}{P_{i,j}}\right| \leq \text{RERR}, \; i=1,2,\ldots,M$$

WS(L) real, working storage array for MLSQQ. Its dimension, L, must be at least N+M+MU(MU+5)/2 where N and M are as above and MU is the number of varying parameters.

IER integer output variable to return from MLSQQ the error flag value, as defined in the EXIT section.

ADDITIONAL ENTRY INFORMATION

(1) Optional additional parameters for MLSQQ are contained in labelled common block MLSQOPT as described below:

/MLSQOPT/LIST,NITER,<u>IWT</u>

LIST integer input variable to control printing.

LIST=0 Only error messages will be printed.

LIST=1 In addition to error messages, parameter values, weighted sum of squares, of residuals, and Marquardt parameter will be printed at each iteration at full printer width.

LIST=2 Same as LIST=1, except line width of printing is suitable for terminal viewing.

LIST=3 Output as with LIST=1 plus diagnostic dump providing normal matrix, right-hand side, lower triangular matrix and its inverse is also printed at each iteration.

LIST=4 At each iteration, the Jacobian is printed in addition to the LIST=3 dump.

Default value is LIST=1 for first call to MSLQQ and previous value thereafter.

NITER integer input variable specifying the maximum number of iterations. Default value is 100.

IWT integer output variable specifying whether a weighted (IWT=1) or unweighted (IWT=0) fit was assumed to calculate the error matrix. An unweighted fit is assumed if all weights W(i) are equal. Otherwise, a weighted fit is assumed.

(2) Auxiliary subroutine, AUX, written by the user, is called by MLSQQ with calling sequence dependent on IOPT value.

If IOPT = 0 or -2, the call is

CALL AUX(X,NRX,F,P,N,IERAUX) (standard form)

If IOPT=±1, the call is

CALL AUX(X,NRX,F,P,N,E,NE,IERAUX) (derivative form)

Since the storage for arrays used in AUX is allocated in the main program, one of the following dimension statements will suffice in AUX:

REAL X(NRX,1),F(1),P(1) for standard form
and
REAL X(NRX,1),F(1),P(1),E(NE,1) for derivative form

X(NRX,NCX) real two dimensional input array of independent variable values as defined for MLSQQ. NCX = number of independent variables.

NRX number of rows of X as dimensioned in the calling program.

F(N) real output array of function values defined by this subroutine.

P(M) real input array of current parameter values passed to AUX from MLSQQ.

N integer input variable specifying number of data points as defined for MLSQQ.

E(N,M) real output array of negative partial derivatives, as defined for MLSQQ

$$E(i,j) = -\frac{\partial f(x_i,p)}{\partial p_j} \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,m \end{matrix}$$

where f is the fitting function. (Note that only partial derivatives with respect to varying parameters are used by MLSQQ so E(i,k) i=1,...,n does not need to be calculated if p_k is fixed.)

NE integer input variable supplying the number of rows of E as dimensioned in the user's program.

IERAUX integer output variable which may be used to pass a user set error flag to MLSQQ. Any non zero value returned in IERAUX will force fatal error termination of LSQQ. The value of the MLSQQ error flag IER is affected by the value of IERAUX as follows:

IERAUX	IER
0 or not defined in AUX	Unchanged
≤ 10	10
> 10	Same as IERAUX

ROUTINES CALLED — AUX supplied by the user, and PRESET, ALERPR and TRACEB from AELIB.

COMMON BLOCKS USED — MLSQOPT

STORAGE REQUIRED — 2200_8

EXIT — There are two possible exits from MLSQQ. These are:

(1) Normal Exit (0 ≤ IER ≤ 2)

Control is returned to the calling routine with fitted parameter values in P, fitted function values in YF, (weighted) residual sum of squares in S and error matrix in E.

(2) Error Exit (IER ≥ 3)

Control is returned to the calling routine with the last approximation to the parameters in P. YF and S are set to contain indefinites.

If no fatal errors have occurred, MLSQQ will terminate by a normal exit. Possible values of IER and associated errors are described below:

IER — Description

0 No errors.

1 Non-fatal error. Value of IOPT supplied by MLSQQ was not 0, 1, -1, or -2. The message

***INVALID OPTION SELECTED - IOPT=0 ASSUMED

is printed, IOPT is set to zero and MLSQQ is allowed to continue.

2 Non-fatal error. Convergence has not been achieved in specified number of iterations. The message

***NO CONVERGENCE AFTER _ ITERATIONS. MAXIMUM RELATIVE CHANGE IN PARAMETERS IS ___.

is printed and MLSQQ is terminated as if convergence achieved.

3 Fatal error. Weights supplied to MLSQQ are not positive. The message

```
***WEIGHTS ARE ZERO, NEGATIVE OR UNDEFINED
```

is printed, E is set to contain indefinites, and MLSQQ is terminated via error exit.

4 Fatal error. Least squares normal system is not numerically positive definite. The message

```
***TRIANGULAR DECOMPOSITION REQUIRES SQUARE ROOT
OF __ IN ROW __.
```

is printed and MLSQQ is terminated via error exit.

≥10 Fatal error. User has specified termination from AUX. E is set to contain indefinites and MLSQQ is terminated via error exit.

ACCURACY

The accuracy of the parameter values fitted by MLSQQ depends directly on the relative error criterion specified by the user. (Whether these fitted parameters are correct depends both on the performance of the least squares algorithm on the function specified and on the starting values supplied by the user. It is left to the user to ensure that the minimum sum of squares of residuals has indeed been achieved and that this is an appropriate (local) minimum for the problem being solved.)

TIMING

The execution time for MLSQQ depends directly on the number of data points, parameters being fitted, and iterations performed as well as on the calculation time for the user's fitting function.

PROGRAM EXAMPLE

A weighted least squares fit of the non-linear function

$$y = e^{p_1+p_2x}$$

to N data points (x,y) is desired. (N is variable but assumed to be less than 100.)

The values for x,y and standard deviation of y, sd(y), are provided on punched cards with one set of x,y,sd(y) values per card.

For this problem MLSQQ is used with IOPT=1 since the derivatives are easily calculated and provided in AUX. The AELIB routine YFST is used with the IMSL routine MDSTI to calculate confidence intervals for function values fitted by MLSQQ.

```
      PROGRAM FIT(INPUT,OUTPUT,TAPE1=INPUT)
      REAL X(100,1),Y(100),W(100),SD(100),YF(100),P(2),E(100,2)
      REAL SDYF(100),CINT(100),WS(210)
      INTEGER NC(2)
      EXTERNAL AUX
 401  FORMAT(3F15.8)
 402  FORMAT(" PREMATURE TERMINATION FROM AUXILIARY ROUTINE ")
 403  FORMAT(" ERROR MATRIX")
 404  FORMAT(1X,2G12.5)
 405  FORMAT("0      X             Y             YFIT          SDYF     CC
     ,INT")
 406  FORMAT(1X,4(G12.5,1X),1H(, G12.5,1H,, G12.5, 1H))
C
C     SET UP PROBLEM DEPENDENT PARAMETERS FOR MLSQQ
C
      DATA NRX/100/,NC/2*1/,M/2/,NE/100/
C
C     READ IN DATA AND SET WEIGHTS
C
      N=1
 40   READ 401,X(N,1),Y(N),SD(N)
      IF(EOF(1).NE.0) GOTO 50
         W(N)=1./SD(N)**2
         N=N+1
         GOTO 40
 50   N=N-1
C
C     SET UP CASE DEPENDENT PARAMETERS FOR MLSQQ AND CALL MLSQQ
C
      IOPT=1
      RERR=1.E-3
      P(1)=15.
      P(2)=-.36E-3
      CALL MLSQQ(IOPT,X,NRX,Y,W,YF,P,NC,N,M,AUX,E,NE,S,RERR,WS,IER)
C
C     CHECK ERROR FLAG FOR EXIT FROM AUX
C
      IF(IER.NE.10) GOTO 75
         PRINT 402
         GOTO 900
```

```
C
C     PRINT ERROR MATRIX FOR FITTED PARAMETERS.(FINAL PARAMETER
C     VALUES AND SUM OF SQUARES HAVE BEEN PRINTED BY MLSQQ)
 75   PRINT 403
C
      DO 1ØØ I=1,M
         PRINT 4Ø4,(E(I,J),J=1,M)
 1ØØ  CONTINUE
C
C     USING YFST TO PROVIDE STANDARD DEVIATION ESTIMATES, CALCULATE
C     95 PERCENT CONFIDENCE INTERVAL ABOUT FITTED VALUES.( IMSL ROUTINE
C     MDSTI IS REQUIRED FOR T VALUE)
C     PRINT FIT RESULTS
C
      CALL YFST(IOPT,X,NRX,Y,SDYF,P,NC,N,M,AUX,E,NE,WS,IER)
      Q=.Ø5
      DEG=N-M
      CALL MDSTI(Q, DEG,TVAL)
      PRINT 4Ø5
      DO 15Ø I=1,N
         CINT(I)=TVAL*SDYF(I)
         PRINT 4Ø6,X(I),Y(I),YF(I),SDYF(I),YF(I)-CINT(I),YF(I)+CINT(I)
 15Ø  CONTINUE
C
 9ØØ  STOP
      END

      SUBROUTINE AUX(X,NRX,F,P,N,E,NE,IERAUX)
C
C     AUXILIARY ROUTINE TO COMPUTE
C             EXP(P1 + P2*X)
C     NOTE -
C        THE NUMBER OF INDEPENDENT VARIABLES IS IMPLICITLY 1.
C        THE ERROR FLAG IS USED TO TERMINATE GRACEFULLY FROM MLSQQ IF
C             EXPONENTIAL ARGUMENT IS OUT OF RANGE
C
      REAL X(NRX,1),F(1),P(1),E(NE,1)
      DO 1Ø J=1,N
         EARG=(P(1) + P(2)*X(J,1))
         IF((EARG.GT.741.67).O.(EARG.LT.-675.84)) GOTO 2Ø
         EVAL=EXP(EARG)
         E(J,1)=-EVAL
         E(J,2)=-X(J,1)*EVAL
         F(J)=EVAL
 1Ø   CONTINUE
      GOTO 9Ø
C
 2Ø   IERAUX=1Ø
 9Ø   RETURN
      END
```

REFERENCES "Derivative Free Analogues of the Levenberg-Marquardt and Gauss Algorithms for Numerical Least Squares Approximation", K.M. Brown, J.E. Dennis, Numerische Mathematik, 18, 289-297.

AUTHOR AND REVISION HISTORY

Adapted from LSQQ by L. Evans and E. Long, April 1977.

Original LSQQ written by J. Schmidt, June 1973, and revised by L. Evans, April 1974.

TITLE

Calculation of Standard Deviation Estimates for the Function Values Fitted by MLSQQ.

INTRODUCTION

The model fitted by MLSQQ has an associated variance as expressed in the error matrix for the fitted parameters. YFST expresses this variance in terms of the fitted function values by calculating a standard deviation estimate, σ_i, for each fitted value YF_i.

A common practice is to calculate confidence intervals for a predicted function value, Y_i. For this the standard deviation estimate, σ_i' should be used where

$$\sigma_i' = \begin{cases} \sqrt{\sigma_i^2 + S/N\text{-}M} & \text{if MLSQQ fit was unweighted} \\ \sqrt{\sigma_i^2 + 1/W(i)} & \text{if MLSQQ fit was weighted} \end{cases}$$

where S, N, M, W(i), i=1,...,N are as defined for MLSQQ.

ENTRY

CALL YFST(IOPT,X,NRX,Y,SDY,P,NC,N,M,AUX,E,NE,WS,IER)

IOPT — integer input variable to specify whether or not derivatives are provided in AUX. If |IOPT| = 1, YFST assumes derivatives are calculated by AUX. For any other value of IOPT, YFST approximates derivatives. (This is consistent with the IOPT parameter used by MLSQQ.)

X(NRX,NCX) — real input array of values of the independent variables as defined for MLSQQ.

NRX — number of rows of X as dimensioned in the calling program.

Y(N) — real input array of values of the dependent variable.

SDY(N) — real output array of standard deviation estimates for the fitted function values.

P(M) — real input array of parameters fitted by MLSQQ.

NC(M) integer input array specifying whether parameters are fixed or varying. See MLSQQ write-up for details.

N integer, number of data points.

M integer, number of parameters in the fitting function.

AUX subroutine supplied by the user to calculate the fitting function. See MLSQQ write-up for details.

E(NE,NCE) real input array containing the error matrix defined by MLSQQ.

NE number of rows of E as dimensioned in the calling program.

WS(L) real working storage array for YFST, $L \geq M^2 + 2N$.

IER integer output variable to return from YFST the error flag value as defined in the EXIT section below.

ROUTINES CALLED AUX supplied by the user and PRESET from AELIB.

EXIT There are two possible exits from YFST. These are:

(1) Normal Exit ($0 \leq IER \leq 1$)

Control is returned to the calling routine. If IER=0, all N entries in SDY are standard deviation estimates. If IER=1, one or more variance estimates were calculated to be negative. In this case, the corresponding standard deviation estimates are set to contain indefinites.

(2) Fatal Error ($IER \geq 10$)

User has specified termination of YFST from AUX. User termination of YFST from AUX is accomplished in the same way as for MLSQQ and the effect on the YFST error flag is the same. (See MLSQQ write-up for details.) In this case, SDY is set to contain indefinites and control is returned to the calling routine.

ACCURACY

The accuracy of these estimates is difficult to establish. They are intended to be used in a general statistical analysis of the least squares solution, not in subsequent calculations requiring a great deal of precision.

TIMING

The execution time for YFST depends on the number of data points, the number of parameters being fitted and the complexity of the user's function. In general it will be much less than the execution time for MLSQQ.

PROGRAM EXAMPLE

The use of this routine is illustrated in write-up for MLSQQ.

AUTHOR & REVISION HISTORY

L.E. Evans and E.G. Long May 1977

TITLE Weighted Least Squares Fitting of Data to a Function of One or More Independent Variables With or Without User Supplied Partial Derivatives. (Recommended for use over MLSQQ.)

ENTRY CALL NL2INT(IOPT,X,NRX,Y,W,YF,P,NC,N,M,AUX,E,NE,S,RERR, WS,IER)

IOPT integer input variable indicating whether or not derivatives are supplied. If they are not they will be approximated.

0,2 no derivatives supplied
±1 derivatives supplied

X(N,NCX) real input array of values of the independent variables, X(i,j) being the ith data value of the jth independent variable. Note: If there is more than one independent variable, NCX must be defined in COMMON block NL2COM described below.

NRX integer input variable, the number of rows of X as dimensioned in the calling program.

Y(N) real input array of values of the dependent variable.

W(N) real input array of weights for dependent variable. For a weighted fit, estimates of the variances of the dependent variable values should be used as follows:

W(i) = 1/var(Y(i)), i=1,...,N

For an unweighted fit, any non-zero constant value may be used but

W(i) = 1, i=1,...,N

is recommended.

YF(N) real output array of fitted values of dependent variable returned from NL2INT.

P(M) real input and output array of parameters for user supplied function. Initial values are passed to NL2INT and final values returned from NL2INT.

NC(M) integer input array specifying whether parameters are to be fixed or varied. NC(i)=0 means ith parameter fixed and NC(i)=1 means ith parameter varied. MU is the number of nonzero entries in NC which defines the number of parameters to be varied.

N integer, number of data points.

M integer, number of parameters in the fitting function (M < N).

AUX subroutine supplied by the user to calculate the fitting function. It must be declared EXTERNAL in the main program. Its calling sequence and arguments are described below.

E(N,M) real array, used in two different ways by NL2INT:

(1) E(N,M) is used internally by AUX to store the negative of the Jacobian of the user's function at each iteration.

(2) E(M,M) is an output array containing the estimated error matrix for the fitted parameters as follows:

E(i,j)=0 if P(i) and/or P(j) are fixed parameters; otherwise,

$$E(i,j) = \begin{cases} \sqrt{\mathrm{var}(P(i))}, & i=j, i=1,\ldots,m \text{ (diagonal)} \\ \mathrm{cov}(P(i),P(j)), & \begin{matrix} j=2,\ldots,m; \\ i=1,\ldots,j-1 \end{matrix} \text{ (above diagonal)} \\ \dfrac{\mathrm{cov}(P(i),P(j))}{\sqrt{\mathrm{var}(P(i)).\mathrm{var}(P(j))}}, & \begin{matrix} i=2,\ldots,m; \\ j=1,\ldots,i-1 \end{matrix} \text{ (below diagonal)} \end{cases}$$

NE integer, number of rows of E as dimensioned in calling program.

S real output variable to return the weighted residual sum of squares from NL2INT.

$$S = \sum_{i=1}^{N} W(i)(Y(i)-YF(i))^2$$

RERR real input variable specifying relative error necessary for convergence of NL2INT. The program calculates

$$\max_i \left| \frac{P_{i,j}-P_{i,j-1}}{P_{i,j}} \right| \quad i=1,2,\ldots,M$$

where the value of the ith parameter after the jth iteration is $P_{i,j}$. If this value is less than RERR and the decrease in the sum of squares for the jth iteration is at most twice the predicted decrease, then NL2INT has converged.

WS(L) real, working storage array for NL2INT. Its dimension, L, must be at least 93+MU(3MU+35)/2 + N(MU+NCX+6) + M(N+2) where N and M are as above, MU is the number of varying parameters, NCX is the number of independent variables.

IER integer output variable to return from NL2INT the error flag value, as defined in the EXIT section.

ADDITIONAL ENTRY INFORMATION

(1) Optional additional parameters for NL2INT are contained in labelled common blocks MLSQOPT and NL2COM as described below:

/MLSQOPT/LIST,NITER,IWT

LIST integer input variable to control printing.

LIST=0 Only error messages will be printed.

LIST=1,3 In addition to error messages, a summary line is printed at each iteration in the form of a table with:

Heading	Meaning
IT	Iteration number.
NF	Number of calls to AUX specifically to evaluate the function (i.e. not concerned with the partial derivatives).
F	0.5*residual sum of squares.
RELDF	Relative difference between previous and current residual sum of squares.
PRELDF	Predicted value of RELDF from quadratic model.
RELDX	Half the relative change in P caused by step just taken.
MODEL	Code indicating models used in choosing current step (G=Gauss-Newton model, S=augmented model).
STPPAR	Marquardt parameter.
S	Sizing factor.
D*STEP	2-norm of the scale vector (D) times the step just taken.
NPRELDF	> 0 Predicted value of RELDF from full Newton step. = 0 Hessian approximation not positive definite. < 0 Negative of predicted value of RELDF for maximum allowable step.

An informative message about the reason for termination of the iterations is also printed.

LIST=2 Same as LIST=1,3 except line width of printing is suitable for terminal viewing and only the first 6 items listed above are printed.

LIST=4 In addition to the information provided if LIST=1,3 the following are printed: the initial values of the varying parameters and the scale vector (under the heading INITIAL X(I) D(I)); the final values of the varying parameters, the scale vector, and the gradient vector (under the heading FINAL X(I) D(I) G(I)); summary statistics such as half the final residual sum of squares, the final values of RELDX, PRELDX, and NPRELDX; and the number of times AUX was called to calculate function values and to calculate partial derivatives excluding the calculation of the variance-covariance matrix and the number of times AUX was called to calculate the calculation of the variance-covariance matrix; and the variance-covariance matrix.

Default value is LIST=1 for first call to NL2INT and previous value thereafter.

NITER integer input variable specifying the maximum number of iterations. Default value is 100.

IWT Integer output variable specifying whether a weighted (IWT=1) or unweighted (IWT=0) fit was assumed to calculate the error matrix. An unweighted fit is assumed if all weights W(i) are equal. Otherwise, a weighted fit is assumed.

/NL2COM/ NCX,IV(100)

NCX integer input variable specifying the number of independent variables. Default value is 1 for the first call to NL2INT and previous value thereafter.

IV integer working storage for NL2INT. (If more than 40 parameters are being varied the source code of NL2INT will need to be recompiled with NIV(60+MU).)

(2) Auxiliary subroutine, AUX, written by the user, is called by NL2INT with calling sequence dependent on IOPT value.

If IOPT = 0 or -2, the call is

```
CALL AUX(X,NRX,F,P,N,IERAUX)   (standard form)
```

If IOPT = ± 1, the call is

```
CALL AUX(X,NRX,F,P,N,E,NE,IERAUX)  (derivative form)
```

Since the storage for arrays used in AUX is allocated in the main program, one of the following dimension statements is required in AUX:

```
REAL X(NRX,1),F(1),P(1) for standard form, and
REAL X(NRX,1),F(1),P(1),E(NE,1) for derivative form
```

X(NRX,NCX) real two dimensional input array of independent variable values as defined for NL2INT. NCX = number of independent variables.

NRX integer variable specifying the number of rows of X as passed to AUX. This value may or may not equal the number of rows of X as dimensioned in the user's calling program. However, it is important that X be dimensioned as X(NRX,1) since X may or may not be the actual array X as defined in the user's calling program.

F(N) real output array of function values defined by this subroutine.

P(M) real input array of current parameter values passed to AUX from NL2INT.

N integer input variable specifying number of data points as defined for NL2INT.

E(NE,M) real output array of negative partial derivatives, as defined for NL2INT

$$E(i,j) = -\frac{\partial f(x_i,p)}{\partial p_j} \qquad \begin{array}{l} i=1,\ldots,n \\ j=1,\ldots,m \end{array}$$

where f is the fitting function. (Note that only partial derivatives with respect to varying parameters are used by NL2INT so E(i,k) i=1,...,n does not need to be calculated if p_k is fixed.)

NE integer variable specifying the number of rows of E as passed to AUX. This value may or may not equal the number of rows of E as dimensioned in the user's calling program. However, it is important that E be dimensioned E(NE,1) since E may or may not be the actual array E as defined in the user's calling program.

IERAUX integer output variable which may be used to pass a user set error flag to NL2INT. A nonzero value returned in IERAUX may cause the calculation of a more acceptable set of parameter values to be attempted or it may cause fatal error termination of NL2INT. The value of the NL2INT error flag IER is affected by the value of IERAUX as follows:

IERAUX	IER
0 or not defined in AUX	Unchanged
≤ 10	10
> 10	Same as IERAUX

ROUTINES CALLED AUX supplied by the user, and NL2SOL, NL2SNO, SETBAD, PRESET, ALERPR and TRACEB from AELIB. Utility routines loaded with NL2INT: CALCJ, CALCR.

COMMON BLOCKS USED MLSQOPT, NL2COM

EXIT There are two possible exits from NL2INT. These are:

(1) Normal Exit ($0 \leq \text{IER} \leq 4$)

Control is returned to the calling routine with fitted parameter values in P, fitted function values in YF, (weighted) residual sum of squares in S and, usually, error matrix in E.

(2) Error Exit ($\text{IER} \geq 3$)

Control is returned to the calling routine with the last approximation to the parameters in P. YF, S, and E are set to contain indefinites.

If no fatal errors have occurred, NL2INT will terminate by a normal exit. Possible values of IER and associated errors are described below:

IER	Description
0	No errors.
1	Non-fatal error. Value of IOPT supplied by NL2INT was not 0, 1, -1, or -2. The message

***INVALID OPTION SELECTED - IOPT=0 ASSUMED is printed, IOPT is set to zero and NL2INT is allowed to continue.

2 Non-fatal error. Convergence has not been achieved in specified number of iterations. The message

```
***NO CONVERGENCE AFTER _ ITERATIONS.
```

is printed and NL2INT is terminated as if convergence was achieved except that the variance-covariance matrix will not be returned and the message for IER=3 will be printed.

3 Non-fatal error. The variance-covariance matrix could not be calculated. This is either due to non-convergence or because in the calculations it was found that the model contained too many parameters. The message

```
*** VARIANCE-COVARIANCE MATRIX NOT CALCULATED
```

is printed, E is set to contain indefinites, and NL2INT is terminated normally.

4 Non-fatal error. The number of non-varying parameters is 0 (i.e. all elements of the array NC are 0). The message

```
*** NO VARYING PARAMETERS SPECIFIED
```

is printed and NL2SOL is terminated normally except that E is set to contain indefinites and the message for IER=3 is also printed.

5 Fatal error. Weights supplied to NL2INT are not positive. The message

```
*** WEIGHTS ARE ZERO, NEGATIVE OR UNDEFINED
```

is printed and NL2INT is terminated via error exit.

6 Fatal error. Value of N not positive or M < N. The message

```
*** VALUE OF N OR M IS ILLEGAL  N= _ M = _
```

is printed and NL2INT is terminated via error exit.

7 Singular convergence. The current value of the residual sum of squares may or may not be minimum. The model breaks down near the current parameter values usually because it has been overspecified (i.e. contains too many parameters). A different starting guess may yield convergence at another set of parameters. The message

```
*** TOO MANY PARAMETERS AT P
```

is printed and NL2INT is terminated via error exit.

8 False convergence. The iterates appear to be converging to a noncritical point. The message

```
*** CONVERGED ON A NONCRITICAL POINT
```

is printed and NL2INT is terminated via error exit.

9 Fatal error. An unexpected internal error has occurred in NL2INT. The message

```
*** INTERNAL ERROR
```

is printed and NL2INT is terminated via error exit.

≥ 10 Fatal error. User has specified termination from AUX from which NL2INT cannot recover. Termination is via error exit.

ACCURACY

The accuracy of the parameter values fitted by NL2INT depends directly on the relative error criterion specified by the user. (Whether these fitted parameters are correct depends both on the performance of the least squares algorithm on the function specified and on the starting values supplied by the user. It is left to the user to ensure that the minimum sum of squares of residuals has indeed been achieved and that this is an appropriate (local) minimum for the problem being solved.)

TIMING

The execution time for NL2INT depends directly on the number of data points, parameters being fitted, and iterations performed as well as on the calculation time for the user's fitting function.

PROGRAM EXAMPLE

A weighted least squares fit of the non-linear function

$$y = e^{p_1+p_2x}$$

to N data points (x,y) is desired. (N is variable but assumed to be less than 100.)

The values for x,y and standard deviation of y, sd(y), are provided on punched cards with one set of x,y,sd(y) values per card.

For this problem NL2INT is used with IOPT=1 since the derivatives are easily calculated and provided in AUX.

```
      PROGRAM FIT(INPUT,OUTPUT,TAPE1=INPUT)
      REAL X(100,1),Y(100),W(100),SD(100),YF(100),P(2),E(100,2)
      REAL SDYF(100),CINT(100),WS(1201)
      INTEGER NC(2)
      EXTERNAL AUX
  401 FORMAT(3F15.8)
  402 FORMAT(" PREMATURE TERMINATION FROM AUXILIARY ROUTINE ")
  403 FORMAT(" ERROR MATRIX")
  404 FORMAT(1X,2G12.5)
  405 FORMAT("0      X           Y          YFIT         SDYF     CONF.
     ,INT")
  406 FORMAT(1X,4(G12.5,1X),1H(, G12.5,1H,, G12.5, 1H))
C
C     SET UP PROBLEM DEPENDENT PARAMETERS FOR NL2INT
C
      DATA NRX/100/,NC/2*1/,M/2/,NE/100/
C
C     READ IN DATA AND SET WEIGHTS
C
      N=1
   40 READ 401,X(N,1),Y(N),SD(N)
      IF(EOF(1).NE.0) GOTO 50
         W(N)=1./SD(N)**2
         N=N+1
         GOTO 40
   50 N=N-1
C
C     SET UP CASE DEPENDENT PARAMETERS FOR NL2INT AND CALL NL2INT
C
```

```
      IOPT=1
      RERR=1.E-3
      P(1)=15.
      P(2)=-.36E-3
      CALL NL2INT(IOPT,X,NRX,Y,W,YF,P,NC,N,M,AUX,E,NE,S,RERR,WS,IER)
C
C     CHECK ERROR FLAG FOR EXIT FROM AUX
C
      IF(IER.NE.10) GOTO 75
         PRINT 402
         GOTO 900
C
C     PRINT ERROR MATRIX FOR FITTED PARAMETERS.(FINAL PARAMETER
C     VALUES AND SUM OF SQUARES HAVE BEEN PRINTED BY NL2INT)
 75   PRINT 403
C
      DO 100 I=1,M
         PRINT 404,(E(I,J),J=1,M)
 100  CONTINUE
C
C     USING YFST TO PROVIDE STANDARD DEVIATION ESTIMATES, CALCULATE
C     95 PERCENT CONFIDENCE INTERVAL ABOUT FITTED VALUES.( IMSL ROUTINE
C     MDSTI IS REQUIRED FOR T VALUE)
C     PRINT FIT RESULTS
C
      CALL YFST(IOPT,X,NRX,Y,SDYF,P,NC,N,M,AUX,E,NE,WS,IER)
      Q=.05
      DEG=N-M
      CALL MDSTI(Q, DEG,TVAL)
      PRINT 405
      DO 150 I=1,N
         CINT(I)=TVAL*SDYF(I)
         PRINT 406,X(I),Y(I),YF(I),SDYF(I),YF(I)-CINT(I),YF(I)+CINT(I)
 150  CONTINUE
C
 900  STOP
      END
      SUBROUTINE AUX(X,NRX,F,P,N,E,NE,IERAUX)
C
C     AUXILIARY ROUTINE TO COMPUTE
C             EXP(P1 + P2*X)
C     NOTE -
C        THE NUMBER OF INDEPENDENT VARIABLES IS IMPLICITLY 1.
C        THE ERROR FLAG IS USED TO TERMINATE GRACEFULLY FROM MLSQQ IF
C             EXPONENTIAL ARGUMENT IS OUT OF RANGE
C
```

```
      REAL X(N,1),F(1),P(1),E(N,1)
      DO 10 J=1,N
         EARG=(P(1) + P(2)*X(J,1))
         IF((EARG.GT.741.67).O.(EARG.LT.-675.84)) GOTO 20
         EVAL=EXP(EARG)
         E(J,1)=-EVAL
         E(J,2)=-X(J,1)*EVAL
         F(J)=EVAL
 10   CONTINUE
      GOTO 90
C
 20   IERAUX=10
 90   RETURN
      END
```

REFERENCES

[1] "An Adaptive Nonlinear Least-Squares Algorithm," John E. Dennis, Jr., David M. Gay, and Roy E. Welsch, TOMS, Vol. 7, pp 348-368.

[2] Algorithm 573, "NL2SOL - An Adaptive Nonlinear Least-Squares Algorithm," John E. Dennis, Jr., David M. Gay, and Roy E. Welsch, TOMS, Vol. 7, pp 369-383.

AUTHOR AND REVISION HISTORY

M. Kent	1981 August
Revised by J. Fisher and C.A. Wills	1982 May

TITLE Minimize the sum-of-squares function of a continuously differentiable function (residual vector) $R(x)=(R_1(x),R_2(x),\ldots,R_n(x))^T$ of p parameters $x=(x_1,x_2,\ldots,x_p)^T$ with or without user supplied partial derivatives.

ENTRY If partial derivatives are being supplied, the call is

CALL NL2SOL(N,P,$\underset{\sim}{X}$,CALCR,CALCJ,$\underset{\sim}{IV}$,$\underset{\sim}{V}$,UIPARM,URPARM,UFPARM)

If partial derivatives are not being supplied the call is

CALL NL2SNO(N,P,$\underset{\sim}{X}$,CALCR,$\underset{\sim}{IV}$,$\underset{\sim}{V}$,UIPARM,URPARM,UFPARM)

N integer input variable, the number of elements in the residual vector R.

P integer input variable, the number of parameters on which R depends.

X(P) real input and output array of parameters. Initial values are passed to NL2SOL or NL2SNO and the best estimates found so far are returned from NL2SOL or NL2SNO.

CALCR subroutine supplied by the user to calculate the residual vector R given values of the parameters. It must be declared EXTERNAL in the calling program. Its calling sequence and arguments are described below.

CALCJ subroutine supplied by the user to calculate the Jacobian matrix J of first partial derivatives given values of the parameters. It must be declared EXTERNAL in the calling program. Its calling sequence and arguments are described below.

IV integer working storage array of length P+60. On input, IV(1) should be set to 0 unless the user is changing certain default values (see section 3 below for more details).

V real working storage array of length 93+N(P+3)+P(3P+33)/2. Its contents are described in section 3 below.

UIPARM, URPARM, UFPARM — these may be real or integer variables, real or integer arrays, or subroutine names. They are passed without change to CALCR and CALCJ where they may be used in the calculations. If one or more subroutines are passed, they must be declared EXTERNAL in the calling program.

ADDITIONAL ENTRY INFORMATION

(1) The subroutine CALCR, written by the user to calculate the residual vector, is called by NL2SOL or NL2SNO with the following calling sequence:

CALL CALCR (N,P,X,NF,R,UIPARM,URPARM,UFPARM)

N — integer input variable, the number of elements in the residual vector R.

P — integer input variable, the number of parameters on which R depends.

X(P) — real input array of current parameter values.

NF — integer input and output variable which indicates whether the residual vector R was properly calculated at the current parameter values. If valid values of R are returned, NF should not be changed. If R cannot be calculated set NF to 0. (This causes NL2SOL or NL2SNO to try to calculate new parameter values.)

R — real output array of values of the residual vector.

UIPARM, URPARM, UFPARM — real or integer variables, real or integer arrays, or subroutine names as defined by the user when calling NL2SOL or NL2SNO.

(2) The subroutine CALCJ, written by the user to supply partial derivatives, is called by NL2SOL with the following statement:

CALL CALCJ (N,P,X,NF,J,UIPARM,URPARM,UFPARM)

N — integer input variable, the number of elements in the residual vector R.

P — integer input variable, the number of parameters on which R depends.

X(P) real input array of current parameter values.

NF integer input and output variable which indicates whether the Jacobian matrix J was properly calculated at the current parameter values. If valid values of J are returned, NF should not be changed. If J cannot be calculated, set NF to 0. This will force termination of NL2SOL.

J(N,P) real output array of partial derivatives defined as:

$$J(i,j) = \frac{\partial R_i(x)}{\partial x_j} \qquad \begin{array}{l} i = 1,2,\ldots,n \\ j = 1,2,\ldots,p \end{array}$$

J must be declared to have exactly N rows, as, for example:

```
REAL J(N,1)
```

UIPARM, URPARM, UFPARM real or integer variables, real or integer arrays, or subroutine names as defined by the user when calling NL2SOL.

(3) On input to NL2SOL and NL2SNO, the IV and V arrays contain certain values that control the behaviour of NL2SOL or NL2SNO. Setting IV(1) = 0 ensures that default values are supplied by NL2SOL or NL2SNO. These default values include:

(a) the maximum number of interations allowed is 150.

(b) the maximum number of function values allowed in 200.

(c) the covariance matrix is computed using the formula

$$V = \sigma^2 H^{-1} (J^T J) H^{-1}$$

where H is a finite-difference approximation to the Hessian and σ^2 is the estimated variance about the regression.

(d) the initial and final x values are printed, a summary line is printed after every iteration (including the sum of squares), summary statistics are printed just before returning to the calling program, and the covariance matrix is printed at the solution.

(e) the absolute function convergence tolerance is 10^{-20}.

(f) the relative function convergence tolerance is about 7.1×10^{-10}.

(g) the X-convergence tolerance is about 8.4×10^{-8}.

(h) the false convergence tolerance is about 7.1×10^{-13}.

These values, and others, may be changed.

To supply nondefault values for selected components of IV and V, the user should first set up these arrays with their default values by

```
     CALL DFAULT(IV,V)
```

and then assign the appropriate nondefault values before calling NL2SOL or NL2SNO. See reference [2] for a description of the valid values for each component of IV and V.

One change from reference [2] should be noted. IV(21) does not contain the output unit number in this version. All printing is automatically done on the file OUTPUT. However, setting IV(21)=0 still suppresses all printing.

(4) On output the IV and V arrays contain information about the solution of the problem. These include,

(a) IV(1) is the return code with values as described in the EXIT section.

(b) IV(26) indicates whether a covariance matrix was computed. If IV(26) > 0, then the lower triangle of the covariance matrix is stored row-wise in V, starting at V(IV(26)). If IV(26)<0, then no covariance matrix was computed.

(c) IV(50) is the starting subscript in V of the residual vector R corresponding to the final X.

(d) IV(10) is the index in V of half the residual sum of squares at the current X.

$$V(IV(10)) = \frac{1}{2} \sum_{i=1}^{n} R_i(X)^2$$

Many other values are also available and these are described in reference [2].

ROUTINES CALLED: CALCJ, CALCR, and up to 3 additional subroutines supplied by the user as well as ALERPR and TRACEB from AELIB. Utility routines loaded with NL2SOL/NL2SNO: ASSESS, COVCLC, DFAULT, DOTPRD, DUPDAT, GQTSTP, ITSMRY, LINVRT, LITVMU, LIVMUL, LMSTEP, LSQRT, LSVMIN, LTSQAR, NL2ITR, PARCHK, QAPPLY, QRFACT, RELDST, RPTMUL, SLUPDT, SLVMUL, STOPX, VAXPY, VCOPY, VSCOPY, V2NORM.

STORAGE REQUIRED: $16,302_8$

EXIT: There are three possible exits from NL2SOL. These are:

(1) Normal Exit ($3 \le IV(1) \le 6$)

Control is returned to the calling routine with the final parameter values in X and the residual vector, half the residual sum of squares, and covariance matrix starting in V(IV(50)), V(IV(10)) and V(IV(26)) respectively.

(2) Exit on non-optimal parameter values ($7 \le IV(1) \le 10$)

Control is returned to the calling routine with the best parameter values found so far in X and the current residual vector and half the residual sum of squares in V(IV(50)) and V(IV(10)) respectively. The covariance matrix is not calculated.

(3) Error Exit ($IV(1) \ge 11$)

Control is returned to the calling program with no guarantee as to the contents of the X, IV, or V arrays.

The user should test IV(1) before continuing.

Some of the possible values of IV(1) are described below. See reference [2] for a complete list of return codes.

(1) $3 \leq IV(1) \leq 6$ (Normal Exit)

3 X convergence. The scaled relative difference between the current parameter vector X and the true locally optimal parameter x* is very likely at most V(33). The message

***** X-CONVERGENCE *****

is printed and control is returned to the calling routine.

4 relative function convergence. The relative difference between the current function value and its locally optimal value is very likely at most V(32). The message

***** RELATIVE FUNCTION CONVERGENCE *****

is printed and control is returned to the calling routine.

5 both X and relative function convergence. The conditions for IV(1)=3 and IV(1)=4 both hold. The message

***** X- AND RELATIVE FUNCTION CONVERGENCE *****

is printed and control is returned to the calling routine.

6 absolute function convergence. One half the current sum of squares is at most (V(31). The message

***** ABSOLUTE FUNCTION CONVERGENCE *****

is printed and control is returned to the calling routine.

(2) <u>7 ⩽ IV(1) ⩽ 10 (Exit on non-optimal parameter values)</u>

7 singular convergence. The Hessian near the current X appears to be singular or nearly so and relative function convergence has occurred. This means that the model is over-specified (i.e. contains too many parameters), at least near X. It is possible that a different starting guess would lead NL2SOL or NL2SNO to find an X giving a smaller sum of squares and strong convergence (IV(1)=3,4,5,or 6). The message

***** SINGULAR CONVERGENCE *****

is printed and control is returned to the calling routine with the best estimates found so far.

8 false convergence. The iterates appear to be converging to a noncritical point. This may mean that the false convergence tolerance V(34) is too large, that the convergence tolerances V(31), V(32), and V(33) are too small for the accuracy to which CALCR and CALCJ compute R and J, that there is an error in computing the Jacobian matrix, or that R is discontinuous near X. The message

***** FALSE CONVERGENCE *****

is printed and control is returned to the calling routine with information about the noncritical point found.

9 function evaluation limit reached without other convergence. The message

```
***** FUNCTION EVALUATION LIMIT *****
```

is printed and control is returned to the calling routine with the best estimates found so far.

10 iteration limit reached without other convergence. The message

```
***** ITERATION LIMIT *****
```

is printed and control is returned to the calling routine with the best estimates found so far.

(3) IV(1) ⩾ 11 (Error Exit)

13 F(X) cannot be calculated at the initial X. The message

```
***** INITIAL SUM OF SQUARES OVERFLOWS *****
```

is printed and control is returned to the calling program with the initial parameter estimates.

15 the Jacobian could not be computed at X. The message

```
***** J COULD NOT BE COMPUTED *****
```

is printed and control is returned to the calling routine with the best estimates found so far.

16 N or P out of range (P<O or N<P). The message

```
///// BAD NN, N, OR P...NN=__, N=__, P=__
```

is printed and control is returned to the calling routine with the initial parameter estimates.

ACCURACY The accuracy of the parameter values returned by NL2SOL or NL2SNO depends directly on the convergence tolerances used in the calculations. It is recommended that the user execute the subroutine with the various default values and then consult the references listed below if changes are needed.

TIMING The execution time for NL2SOL or NL2SNO depends directly on the number of elements in the residual vector, parameters on which the residual vector depends, and iterations performed as well as on the calculation time for the user supplied functions CALCR and CALCJ.

The time the program example takes to execute with N = 9 is approximately 0.064 seconds (CYBER 170, Model 175 , 1982 May).

PROGRAM EXAMPLE

The sum-of-squares function to be minimized is

$$F(p) = \sum_{i=1}^{n} \left(\sqrt{(x_i - p_1)^2 + (y_i - p_2)^2} - p_3 \right)^2$$

The n values (x_i, y_i) are provided as data and read in with one point on each card image. Since the derivatives of this function are easily calculated the subroutine NL2SOL is called.

A sample output from NL2SOL is provided with a short explanation.

```
      PROGRAM TEST(INPUT, OUTPUT, TAPE5 = INPUT)
C
      REAL P(3), V(456), X(50), Y(50)
      INTEGER IV(63)
C
      EXTERNAL CALCJ, CALCR
C
C READ N, THE NUMBER OF DATA POINTS (ASSUMED TO BE NO MORE THAN 50)
C AND THE N DATA POINTS
C
      READ 100, N
  100 FORMAT(I2)
C
      DO 10 I=1,N
        READ(5, 110) X(I), Y(I)
  110   FORMAT(E8.2, 1X, E8.2)
   10 CONTINUE
C
C READ IN THE INITIAL ESTIMATES FOR THE THREE PARAMETERS
C
      M = 3
C
      DO 12 I=1,M
        READ(5, 110) P(I)
   12 CONTINUE
C
C THIS CALL TO NL2SOL PRINTS THE INITIAL AND FINAL VALUES OF THE
C PARAMETERS, A SHORTENED SUMMARY LINE FOR EACH ITERATION (INCLUDING
C HALF THE RESIDUAL SUM OF SQUARES), AND THE VARIANCE-COVARIANCE
C MATRIX
C
```

```
      CALL DFAULT(IV, V)
      IV(19) = -1
      CALL NL2SOL(N, M, P, CALCR, CALCJ, IV, V, X, Y, UFPARM)
C
C PRINT THE RETURN CODE
C
      PRINT 120, IV(1)
  120 FORMAT(1H1, "IV(1) = ", I2)
C
C PRINT THE ORIGINAL DATA POINTS AND THE RESIDUAL FOR EACH ONE
C
      DO 20 I=1,N
        PRINT 122, X(I), Y(I), V(IV(50) + I - 1)
  122   FORMAT(1H0, 3E15.6)
   20 CONTINUE
C
      STOP
      END

C
C***********************************************************************
C
      SUBROUTINE CALCJ(N, M, P, NF, J, X, Y, UFPARM)
      INTEGER N, M, NF
      REAL J(N,1), P(1), ROOT, X(1), Y(1)
C
      DO 10 I=1,N
        ROOT = (X(I) - P(1)) ** 2 + (Y(I) - P(2)) ** 2
        IF (ROOT .LE. 0) GOTO 20
        J(I,1) = (P(1) - X(I)) / SQRT(ROOT)
        J(I,2) = (P(2) - Y(I)) / SQRT(ROOT)
        J(I,3) = -1.0
   10 CONTINUE
      RETURN
C
C ERROR, ROOT EQUALS ZERO--CANNOT COMPUTE JACOBIAN
C
   20 NF = 0
      RETURN
      END

C
C***********************************************************************
C
      SUBROUTINE CALCR(N, M, P, NF, F, X, Y, UFPARM)
      INTEGER N, M, NF
      REAL F(1), P(1), ROOT, X(1), Y(1)
C
      DO 10 I=1,N
        ROOT = (X(I) - P(1)) ** 2 + (Y(I) - P(2)) ** 2
        F(I) = SQRT(ROOT) - P(3)
   10 CONTINUE
      RETURN
      END
```

```
     I      INITIAL X(I)       D(I)

     1      -.865621E+04      .294E+01
     2       .488061E+04      .575E+00
     3       .912500E+02      .300E+01

    IT     NF       F          RELDF        PRELDF       RELDX

     0      1    .389E+01
     1      2    .282E-01    .993E+00     .993E+00     .169E-03
     2      3    .282E-01    .805E-03     .804E-03     .565E-05
     3      4    .282E-01    .127E-08     .127E-08     .761E-08

 ***** X-CONVERGENCE *****

     I       FINAL X(I)        D(I)            G(I)

     1      -.865319E+04      .295E+01      -.390E-08
     2       .488064E+04      .557E+00       .357E-09
     3       .932866E+02      .300E+01       .390E-08

 COVARIANCE = SCALE * H**-1 * (J**T * J) * H**-1

 ROW  1      .4400E+01
 ROW  2      .3567E-01   .3064E-01
 ROW  3      .4323E+01   .3503E-01   .4248E+01
```

HEADING	EXPLANATION
I	parameter number
INITIAL X(I)	initial parameter values
D(I)	scale vector for parameters
IT	iteration number
NF	number of function evaluations excluding any needed for computing the covariance matrix or approximating the Jacobian
F	half the residual sum of squares
RELDF	the relative difference between the previous and current residual sum of squares
PRELDF	the relative difference between the previous residual sum of squares and the current residual sum of squares predicted by the model

HEADING — EXPLANATION

RELDX — the maximum relative change in a parameter value between the previous and current iteration

FINAL X(I) — final parameter values

G(I) — the final value of the gradient vector

COVARIANCE — the formula used to calculate the covariance matrix and the value of the matrix

EXTERNAL VERSION — Source code of the versions of NL2SOL and NL2SNO discussed in the references may be obtained from the AELIB UPDATE program library by using the UPDATE directive

```
*DEFINE  EXT
```

The versions of NL2SOL and NL2SNO which are currently in AELIB are the same as the originals except that:

(a) subroutines DOTPRD and V2NORM have been replaced by faster versions which do not continually check for underflow,

(b) all WRITE statements have been changed to PRINT statements,

(c) all error messages include calls to ALERPR and TRACEB to allow printing to be controlled by the user.

Source code obtained from the AELIB UPDATE program library without the DEFINE directive will be the AELIB version.

REFERENCES

(1) "An Adaptive Nonlinear Least-Squares Algorithm", John E. Dennis, Jr., David M. Gay, and Roy E. Welsch, ACM TOMS, Vol. 7, No. 3, pages 348-368.

(2) "Algorithm 573 NL2SOL - An Adaptive Nonlinear Least-Squares Algorithm", John E. Dennis, Jr., David M. Gay, and Roy E. Welsch, ACM TOMS, Vol. 7, No. 3, pages 369-383.

REVISION HISTORY

Adapted from Algorithm 573 by   Anne Wills   1982 May

TITLE Find the approximate radius and centre of curvature given a set of data points that define the curve.

ENTRY CALL RADCURV (X,Y,N,RERR,Q,P,CINT,WK,WS,IER)

X an input array of length N containing the independent variable. The values must be in order around the curve.

Y an input array of length N containing the corresponding dependent variable values.

N input number of data points ($N > 3$).

RERR real input variable specifying relative accuracy to which the radius and centre of curvature are to be calculated.

Q input variable defining the confidence intervals to be calculated for the parameters defining the curve. Usually set to 0.05 to get 95% confidence intervals.

P Real output array of length 3 containing the final estimates for the centre (h,k) and radius r, of curvature where P(1), P(2) and P(3) are h, k and r respectively.

CINT Real output (3,2) array where each row contains the lower and upper values of the confidence intervals for parameters in P.

WK Double precision work vector, length is 4*N.

WS Real working storage array. Dimension must be at least 156+6N where N is as above.

IER Error parameter. Flag indicates which, if any, errors occurred during execution -- defined in EXIT section.

ROUTINES CALLED IMSL Routines LINV1P, MDSTI, RLFOR, VCVTSF, VTPROF.
AELIB Routines ALERPR, NL2SOL, SETBAD, TRACEB.
Utility Routines loaded with RADCURV: RESID, DERRES.

COMMON BLOCKS USED NL2PRNT

EXITS

Normal exit, IER=0
If no fatal errors have occurred during execution then control is returned to the calling program and estimates for the centre (h,k) and radius of curvature r are passed back in array P where P(1), P(2) and P(3) are h, k and r respectively. The confidence intervals about the parameters in P are also returned in array CINT.

Error exits, IER= 1, 3, 4, 5 or 6

IER=1 Fatal error has occurred. The value for N is invalid. The message

```
***INVALID VALUE FOR N, N=_
```

is printed and RADCURV terminates via error exit.

IER=3 Fatal error has occurred. The data is not in order around the curve. The message

```
***DATA NOT IN ORDER AROUND THE CURVE
```

is printed and RADCURV is terminated via error exit.

IER=4 Fatal error has occurred. There is no curvature in the data. The message

```
***DATA DEFINE A STRAIGHT LINE
```

is printed and RADCURV terminates via error exit.

IER=5 No convergence in routine NL2SOL. The message

```
***PREMATURE TERMINATION FROM NL2SOL, IV(1)=_
```

is printed and RADCURV terminates via error exit except that the best estimates found so far for h, k, and r are returned in P.

IER=6 Fatal error has occurred in RADCURV. An error has occurred which is inexplicable. The message

```
***INTERNAL ERROR IN RADCURV
```

is printed and execution is terminated.

ACCURACY

The accuracy of the radius and centre of curvature fitted by RADCURV depend directly on the relative error criterion specified by the user.

TIMING

The execution time for RADCURV depends directly on the number of data points N, involved.

The time the program example takes to execute with 8 data points is 0.057 seconds (CYBER 170 Model 175, March 1982).

PROGRAM EXAMPLE

The following program finds the approximate centre and radius of curvature and confidence intervals of the curve defined by N data points read in from cards (N is assumed to be no more than 50).

If fatal errors are detected the program is terminated and indefinites are returned in the output parameters. Otherwise, the printed output from this program includes

- the centre of curvature
- the radius of curvature
- a matrix containing the confidence intervals.

```
      PROGRAM CURVE(INPUT, OUTPUT, TAPE5 = INPUT)
      INTEGER I,IER,N
      REAL P(3),CINT(3,2),Q,W(50),WS(360),X(50),Y(50)
      DOUBLE PRECISION WK(200)
      DATA IER/0/
C
C    OBTAIN THE DATA POINTS DEFINING THE CURVE (ASSUMED TO BE NO MORE
C    THAN 50)
C
      READ(5,*)N
C
      DO 20 I=1,N
         READ(5,*)X(I),Y(I)
         PRINT 2600, X(I), Y(I)
   20 CONTINUE
C
C    SET UP PARAMETERS NEEDED FOR SUBROUTINE RADCURV AND CALL IT TO
C    CALCULATE CENTRE AND RADIUS OF CURVATURE
C
      Q = 0.05
      RERR = 1.E-2
      CALL RADCURV(X,Y,N,RERR,Q,P,CINT,WK,WS,IER)
```

```
C
C    CHECK ERROR FLAG FOR ABNORMAL TERMINATION OF RADCURV
C
      IF (IER .EQ. 0) GOTO 60
      PRINT 2700
      GOTO 999
C
C    PRINT PARAMETER VALUES AND CONFIDENCE INTERVALS FOR EACH PARAMETER
C
   60 PRINT 2200,P(1),P(2)
      PRINT 2300,P(3)
      PRINT 2400
      PRINT 2500
      DO 70 I = 1,3
         PRINT 2600,(CINT(I,J),J=1,2)
  70  CONTINUE
C
  999 STOP
C
C     FORMAT STATEMENTS
C
 2200 FORMAT(" THE CENTRE OF CURVATURE IS (",E12.5,",",E12.5")")
 2300 FORMAT(" THE RADIUS OF CURVATURE IS ",E12.5)
 2400 FORMAT("      ")
 2500 FORMAT("   CONFIDENCE INTERVALS")
 2600 FORMAT(1X, 3F12.2)
 2700 FORMAT(" ***ABNORMAL TERMINATION FROM RADCURV")
C
      END
```

AUTHORS J.M. Blair
Jane Fisher
C.A. Wills

1982 April

1-12 DATA SMOOTHING

1-12 DATA SMOOTHING

A common problem in computing is to fit a smooth curve to a set of data. The problem may be one of interpolation, in which the curve passes through the data points, or approximation, in which the curve passes close to, but not necessarily through, the points. Whether interpolation or approximation is appropriate depends in general on the magnitude of the errors in the data. This section deals with approximation, while interpolation is treated in section 1-14.

The subroutines SMOOTH and HDERIV use a cubic spline to approximate the unknown function. The number of knots is increased progressively until the scatter of the data points about the curve ceases to show statistically significant trends. The method, due to M.J.D. Powell and described in [1], appears to be the most accurate smoothing algorithm currently available.

The resulting cubic spline approximation may be differentiated to give approximations to the derivatives of the data. This approach, rather than using the polynomial interpolation formulae in section 1-15, is recommended for "noisy" data, that is for data in which the errors are significant.

Chapter I of [2] also contains a number of useful subroutines for interpolation, smooth interpolation, and approximation of data by cubic splines.

J.M. Blair
1981 April

Revised
1983 June

References

[1] M.J.D. Powell, "Curve Fitting by Splines in One Variable", Chapter 6 of "Numerical Approximation to Functions and Data", J.G. Hayes (Ed), Inst. of Math. & Its Appl., Athlone Press, 1970.

[2] IMSL Library Reference Manual.

TITLE

Smooth Curve Fitting - Calculates a smooth curve s(x) which approximates given weighted data. The output is in a form which allows evaluation of the fitted cubic spline, s(x), or its derivative, s'(x), at any point in the data range. (If fitted function values and derivatives are required only at the data points, use AELIB routine HDERIV instead.)

ENTRY

CALL SMOOTH(M,N,XD,YD,WD,RD,XN,FN,GN,IPRINT,WS,IER)

M integer input variable, the number of data points (see note below *). M must be at least 6.

N integer input variable which should be set to an overestimate of the number of knots in the final fit. (See next section for advice on estimating N.)

If a curve has been fitted, the value of N returned will be the number of knots used in the function s(x).

XD a real input array of length M, containing the abscissae of the data points in strictly increasing order (see note below *).

YD a real input array of length M, containing the ordinates of the data points, (see note below *).

WD a real input array of length M containing the weights, WD(J)=W_j, the weight for the jth data point (see note below *).

For an unweighted fit, use W_j=1, j=1,...,M.

For a weighted fit, use W_j=1/sd_j where sd_j is an estimate of the standard deviation for the ordinate of the jth data point.

In addition, W_j=0 will remove the jth data point from the fit and is recommended in preference to giving a point a small weight relative to other points.

It is also possible to force s(x) to pass near selected data points by using very large weights (10^4 times normal is suggested).

IPRINT
integer input variable specifying the printing required.
IPRINT=0 no printing, except for error messages.
IPRINT>0 prints details of the final fit only.
IPRINT<0 prints details of the fit at every iteration using new knots.

WS a real input array of length at least 7M+11N+6 words, used as working storage.

The following arguments are set by SMOOTH:

RD a real output array of length M which is set by SMOOTH to contain the residuals RD(J)=YD(J)-S(XD(J)), J=1,2,...,M.

XN a real output array of length N, the knot positions, ξ_j, j=1,N, used by the fitting function.

FN a real output array of length N, the values of s(x) at the knots.

GN a real output array of length N, the values of s'(x) at the knots.

IER integer output variable, an error flag with values defined in EXIT section below.

*If data points are deleted by SMOOTH, N,XD,YD,WD will be changed to define the new data set.

ESTIMATING N Since the final fit s(x) is determined automatically, the number of knots, N, is initially unknown; however, the user is responsible for dimensioning arrays depending on N. The following remarks are intended to help the user make an overestimate of the space required:

(1) N must be at least 5.

(2) If there are a large number of data points fairly uniformly distributed over the range XD(1) < X < XD(M) then N=M/2 should be an over-estimate.

(3) If the data are concentrated in a small region of the range, then N should be approximately equal to M.

(4) It may be necessary for N to be greater than M. However, N must not be greater than 2*M.

EXIT

Normal Exit (IER=0)

A smooth curve has been successfully fitted. The residuals at the data points or the spline values at the knots can be used to define s(x) or s'(x) as described in the next section.

Error Exits (IER≠0)

(1) IER=1

The next iteration requires more than N knots. The value returned in N is the number of knots in the last iteration. The residuals for the fit at the M data points are stored in RD and therefore the function values at these points may be recovered from the expression YD(J)-RD(J), J=1,...,M. The arrays XN, FN and GN are set to contain indefinites. The message

```
*** ESTIMATE FOR N IS TOO SMALL - NEXT ITERATION
    REQUIRES "15 KNOTS"
```

is printed with traceback and control is returned to the calling routine.

(2) IER=2

Some data points have been removed to force XD to be strictly increasing. The cubic spline has been fitted and all output arrays are defined but the fit may not be suitable. The message

```
*** SOME DATA POINTS HAVE BEEN DELETED
```

is printed with traceback. N, XD, YD, and WD are changed to define the new data set and control is returned to the calling routine. Function and derivative values may be calculated as described in the next section.

(3) IER=3

Not enough data was supplied to define the spline uniquely. A cubic spline has been fitted and all output arrays are defined but the spline may not be suitable. The message

```
*** INSUFFICIENT DATA TO DEFINE THE SPLINE UNIQUELY -
    RESULTS UNRELIABLE
```

is printed with traceback and control is returned to the calling routine. Function and derivative values may be calculated as described in the next section.

(4) IER=4

Value of N supplied is less than 5 or greater than 2*M. All output arrays are set to contain indefinites, the message

```
*** INITIAL ESTIMATE FOR N IS LESS THAN 5 OR IS GREATER
    THAN ___ (twice the number of data points)
```

is printed with traceback and control is returned to the calling routine.

(5) IER=5

Value of XD(1) is greater than value of XD(M). All output arrays are set to contain indefinites, the message

```
*** ABSCISSA OF FIRST DATA POINT MUST BE LESS THAN
    ABSCISSA OF LAST DATA POINT
```

is printed with traceback and control is returned to the calling routine.

EVALUATION OF THE FITTED FUNCTION, S(X), AND/OR ITS DERIVATIVE S'(X)

If IER = 0, 2 or 3, fitted function values at the data points are given by

```
YD(J)=RD(J), J=1,M
```

Fitted function and derivative values at the knots, ξ_j, are provided in the arrays FN and GN respectively. To evaluate the fitted function or derivative values at any other points, the following interpolation formula may be used:

If $\xi_j < x < \xi_{j+1}$, then

$$s(x) = U_j(x)s(\xi_j) + U_{j+1}(x)s(\xi_{j+1}) + V_j(x)s'(\xi_j) + V_{j+1}(x)s'(\xi_{j+1})$$

where

$$U_j(x) = \left[1 + 2\frac{(x-\xi_j)}{(\xi_{j+1}-\xi_j)}\right]\left(\frac{x-\xi_{j+1}}{\xi_j-\xi_{j+1}}\right)^2$$

$$U_{j+1}(x) = \left[1 + 2\frac{(x-\xi_{j+1})}{(\xi_j-\xi_{j+1})}\right]\left(\frac{x-\xi_j}{\xi_{j+1}-\xi_j}\right)^2$$

$$V_j(x) = \frac{(x-\xi_j)(x-\xi_{j+1})^2}{(\xi_j-\xi_{j+1})^2}$$

$$V_{j+1}(x) = \frac{(x-\xi_{j+1})(x-\xi_j)^2}{(\xi_{j+1}-\xi_j)^2}$$

and

$$s'(x) = U_j'(x)[s(\xi_j)-s(\xi_{j+1})] + V_j'(x)s'(\xi_j) + V_{j+1}'(x)s'(\xi_{j+1})$$

where

$$U_j'(x) = \frac{6(x-\xi_j)(x-\xi_{j+1})}{(\xi_{j+1}-\xi_j)^3}$$

$$V_j'(x) = \frac{(x-\xi_{j+1})[(x-\xi_{j+1}) + 2(x-\xi_j)]}{(\xi_j-\xi_{j+1})^2}$$

$$v_{j+1}'(x) = \frac{(x-\xi_j)[(x-\xi_j) + 2(x-\xi_{j+1})]}{(\xi_{j+1}-\xi_j)^2}$$

ROUTINES CALLED VB06A, a utility routine loaded from AELIB with SMOOTH and ALERPR, TRACEB and LIBSTAT from AELIB.

STORAGE 3000_8

EXAMPLE This example reads in a set of data points, assigns equal weights and makes a call to SMOOTH requesting detailed printout. Fitted function values are calculated and printed at each data point and midway between each pair of data points.

```
      PROGRAM SHOW(INPUT=/8Ø,OUTPUT=/136,TAPE5=INPUT)
C
C PROGRAM SHOW ILLUSTRATES THE INTERPOLATION OF A
C FITTED FUNCTION RETURNED BY SMOOTH
C
      DIMENSION XD(1ØØ),YD(1ØØ),WD(1ØØ),WS(1256),YFIT(2ØØ),X(2ØØ),
     1 RD(1ØØ),XN(5Ø),FN(5Ø),GN(5Ø)
C
C     READ THE DATA POINTS FROM CARDS UNTIL  AN END OF FILE
C     IS REACHED OR 1ØØ CARDS HAVE BEEN READ
C
```

```
      DO 1Ø I=1,1ØØ
      M=I
      WD(I)=1.
      READ(5,*) XD(I),YD(I)
      IF(EOF(5)) 15,1Ø
 1Ø   CONTINUE
      GOTO 17
15    M=M-1
17    CONTINUE
      N = M/2
      IPRINT =-1
      CALL SMOOTH(M,N,XD,YD,WD,RD,XN,FN,GN,IPRINT,WS,IER)
      PRINT*," NUMBER OF KNOTS IS ",N
      IF(IER.NE.Ø) GO TO 9ØØ
C
C     SET UP THE X POINTS AT WHICH THE FITTED FUNCTION IS TO BE
C     EVALUATED
C
      K=Ø
      MM=M-1
      DO 2Ø I=1,MM
      K=K+1
      X(K)=XD(I)
      K=K+1
      X(K)=XD(I)+Ø.5*(XD(I+1)-XD(I))
 2Ø   CONTINUE
      K=K+1
      X(K)=XD(M)
C
C     EVALUATE THE FITTED FUNCTION AT EACH X POINT
C
      J=Ø
      KK=K-1
      DO 3Ø I=1,KK
      IF(X(I).GE.XN(J+1)) J=J+1
      Z1=X(I)-XN(J)
      Z2=X(I)-XN(J+1)
      D1=XN(J+1)-XN(J)
      D2=-D1
      UJ=(1+2*Z1/D1)*(Z2/D2)**2
      UJ1=(1+2*Z2/D2)*(Z1/D1)**2
      VJ=(Z1*Z2**2)/D2**2
      VJ1=(Z2*Z1**2)/D1**2
      YFIT(I)=UJ*FN(J)+UJ1*FN(J+1)+VJ*GN(J)+VJ1*GN(J+1)
      PRINT*,X(I),YFIT(I)
 3Ø   CONTINUE
      YFIT(K)=FN(N)
      PRINT*,X(K),YFIT(K)
      STOP
 9ØØ  CONTINUE
C
C     AT THIS POINT THE ERROR FLAG CAN BE CHECKED TO DETERMINE
C     WHAT ERROR CONDITION PREVAILS AND APPROPRIATE ACTION TAKEN
C     FOR THIS EXAMPLE HOWEVER THE ERROR FLAG IS JUST PRINTED
C
      PRINT*," ERROR FLAG IS ",IER
      STOP
      END
```

AUTHOR Adapted from the Harwell library routine VC03A by B.V. Riff, June 1977.

REVISED BY C.A. Wills DATE November 1979

TITLE

Smooth Curve Fitting and Smoothed Derivative Evaluation.

Given M data points (x_i,y_i), i=1,2,...,M and the corresponding weights w_i, i=1,2,...,M, this subroutine calculates a cubic spline fit s(x) with derivatives s'(x) at each of the data points using the AELIB routine SMOOTH.

ENTRY

CALL HDERIV(M,N,XD,YD,WD,YS,YPRIME,WKA,IER,IPRINT)

M integer input variable, the number of data points. M must be at least 6.

N integer input variable, the estimate of the number of knots in final fit. Use N=max(M/2,5) or make an estimate for N as described in the writeup for SMOOTH. If a curve has been fitted, the value of N returned will be the number of knots used.

XD a real input array of length M, the abscissae of the data points in strictly increasing order.

YD a real input array of length M, the ordinates of the data points.

WD a real input array of length M, the weights, W_j. Use W_j=1 for an unweighted fit and W_j=1/sd_j where sd_j is the standard deviation estimate for YD_j. Special weighting options are described in the writeup for SMOOTH.

YS a real output array of length M, the smoothed function values returned by HDERIV.

YPRIME
a real output array of length M, computed derivatives calculated by HDERIV.

WKA real input working storage array; should be dimensioned at least 7M+14N+6 in the calling program.

IER integer output variable, an error flag with values defined in EXIT section below.

IPRINT
integer input variable specifying the printing required.
IPRINT=0 no printing, except for error messages.
IPRINT>0 prints details of the final fit only.
IPRINT<0 print details of the fit at every iteration using new knots.

EXIT

Normal Exit (IER=0)

On a normal return N contains the number of knots used in the fitting function s(x), and arrays YS, YPRIME contain the smoothed function value and derivative value respectively for each data point.

Error Exit (IER≠0)

All error exits are a direct result of errors encountered in SMOOTH. See the writeup of SMOOTH for further details and for the text of error messages issued.

IER=1 next iteration requires larger number of knots than N specified.
IER=2 some data points which were out of order have been removed, data points must be strictly increasing. (In this case, N, XD, YD, WD will be changed to reflect the new data set.)
IER=3 insufficient data to define the spline uniquely.
IER=4 initial estimate of N must be at least 5.
IER=5 XD(1) must be less than XD(M).
IER=6 number of data points M must be at least 6.

For IER=1, 4, 5, or 6, the arrays YS and YPRIME are set to indefinite.

For IER=2 or 3, the arrays YS and YPRIME contain the best values available.

ROUTINES CALLED

SMOOTH and LIBSTAT from AELIB.

EXAMPLE

This example reads in a set of data points, assigns equal weights and calls HDERIV requesting detailed printout. The function and derivative values are then printed for each data point.

```
      PROGRAM SHOW(INPUT=/80,OUTPUT=/136,TAPE5=INPUT)
C
C PROGRAM SHOW ILLUSTRATES THE USE OF HDERIV
C
      DIMENSION XD(100),YD(100),WD(100),WS(1406),YFIT(100),YPRIME(100)
C
C     READ THE DATA POINTS FROM CARDS UNTILL AN END OF FILE
C     IS REACHED OR 100 CARDS HAVE BEEN READ
C
      DO 10 I=1,100
      M=I
      WD(I)=1.
      READ(5,*) XD(I),YD(I)
      IF(EOF(5)) 15,10
  10  CONTINUE
      GOTO 17
 15   M=M-1
 17   CONTINUE
      N = M/2
      IPRINT =-1
      CALL HDERIV(M,N,XD,YD,WD,YFIT,YPRIME,WS,IER,IPRINT)
      PRINT*," NUMBER OF KNOTS IS ",N
      IF(IER.NE.0) GO TO 900
C
      PRINT*,"    X         Y      FITTED Y      FITTED DERIVATIVE"
      DO 20 I=1,M
      PRINT*,XD(I),YD(I),YFIT(I),YPRIME(I)
 20   CONTINUE

      STOP
 900  CONTINUE
C
C     AT THIS POINT THE ERROR FLAG CAN BE CHECKED TO DETERMINE
C     WHAT ERROR CONDITION PREVAILS AND APPROPRIATE ACTION TAKEN
C     FOR THIS EXAMPLE HOWEVER THE ERROR FLAG IS JUST PRINTED
C
      PRINT*," ERROR FLAG IS ",IER
      STOP
      END
```

AUTHOR B.V. Riff DATE June 1977

REVISED BY C.A. Wills DATE November 1979

1-13 TIME SERIES ANALYSIS

1-13 TIME SERIES ANALYSIS

A time series is a collection of observations made sequentially in time. The intervals between observations may vary, but in most applications, which range from economics to engineering, they are of equal duration.

There are two basic problems in time series analysis. The first is the detection of trends in the data (mean values, variances, cycles, serial correlations), and the second is the spectral (frequency) analysis of the data [1]. In the second type of problem the more obvious trends will usually have been removed, so that the residual data approximates to a stationary random process.

The subroutine FFT (Fast Fourier Transform) [2],[3], is used in spectral analysis, provided the number of observations N satisfies certain requirements. The advantage over ordinary fourier analysis is that while the latter corresponds formally to matrix multiplication and requires N^2 operations, FFT uses trigonometric identities (hence the restriction on N) and requires the order of N log N operations. This results in a considerable saving of machine time if N is large. FFT has also been used with spatial data in the solution of elliptic boundary value problems [4].

A number of routines for time series analysis can be found in Chapter F of the IMSL library [5], and in the MAC/RAN data analysis system [6].

W.N. Selander
1981 April

References

[1] C. Chatfield, "The Analysis of Time Series: Theory and Practice", Chapman and Hall, 1975.

[2] W.N. Gentleman and G. Sande, "Fast Fourier Transforms - For Fun and Profit", Proc. AFIPS Computer Conference, p. 563, 1966.

[3] J.W. Cooley and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol. 19, p.297, 1965.

[4] R.W. Hockney, "A Fast, Direct Solution of Poisson's Equation Using Fourier Analysis", J. ACM, Vol. 12, p.95, 1965.

[5] IMSL Library Reference Manual.

[6] MAC/RAN III, Time Series Data Analysis System.

TITLE Fast Fourier Transform

ENTRY CALL FFT(Y,X,N)

Y complex output array of the Fourier transforms.

X complex input array of the Fourier coefficients.

N integer, number of Fourier coefficients in the array. It must be an integral power of 4; i.e. $N=4^m$: m=3,4,...

EXIT The Fourier transforms are stored in Y. Original contents of X are destroyed. No error checking is done.

ACCURACY 12 significant digits.

SPEED 1.05 N $\log_4$ N ms (6600, February 1970)

AUTHOR P.Y. Wong DATE February 1970

1-14 INTERPOLATION

00-09 UNIVARIATE INTERPOLATION

1-14-00	SPINT	INTERPOLATION OF EQUALLY SPACED DATA
1-14-01	ALAGIN	CUBIC LAGRANGE INTERPOLATION
1-14-02	AIKINT	AITKEN INTERPOLATION

10-19 MULTIVARIATE INTERPOLATION

1-14-10	INPOL	MULTIVARIATE INTERPOLATION

1-14 INTERPOLATION

Given a table of values of a function f(x) at arguments $x_0, x_1, x_2, \ldots,$ the problem in interpolation is to compute f(x) at some other argument x=z. The two most common methods of estimating f(z) use either a Lagrange interpolating polynomial or a cubic spline approximation.

The Lagrange polynomial of degree 2 interpolating f(x) at x_0, x_1, x_2 is given by

$$f_{0,1,2}(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2),$$

with higher degree polynomials defined similarly. This parabola coincides with f(x) at $x=x_0, x_1, x_2$, but not, in general, for any other value of x. If z lies between x_0 and x_2, $f_{0,1,2}(z)$ may be used as an approximation to f(z). The accuracy of this approximation depends on the smoothness of f(x). In general, higher degree polynomials are more accurate approximations than lower degree polynomials, but we cannot predict the error for any particular degree. Thus the main difficulty in interpolation is to know what degree of Lagrange polynomial to use in order to achieve the required accuracy in the interpolated value.

The routines in this chapter all use the Lagrange polynomial for interpolation. SPINT is designed for equally spaced data; it uses a modified fifth degree polynomial in the interior of the argument range, and second degree polynomials in the end regions. ALAGIN accepts unequally spaced data, and uses a third degree polynomial. AIKINT is an adaptive routine, which computes first, second, third, ... degree polynominals, until two successive approximations agree to the specified accuracy or until a maximum of nineteen iterations have been performed. INPOL uses a generalization of the Lagrange polynomial to several independent variables, and requires the user to specify the degree of the polynomial in each dimension.

Chapter I of [1] contains routines for interpolation by cubic splines in one and two dimensions.

J.M. Blair
1983 June

References

[1] IMSL Library Reference Manual.

TITLE Quintic Interpolation of equally spaced data.

ENTRY R=SPINT(Z,Y,N,XO,H)

Z real input variable, point for which value of function is required.

Y real input array providing a table of n function values, f(x), as follows:

$$Y(j) = f(x_{j-1}) = f(x_0 + (j-1)h),\ j=1,2,\ldots,n$$

N integer input variable, the number of elements in Y.

XO real input variable, the value of x for the first table entry in Y.

H real input variable, the x interval used to define the table in Y.

EXIT This function returns the real interpolated value of f(z) to the calling routine. There is no error checking performed.

ACCURACY Depends on the smoothness of f(x) and on the mesh spacing h. For $x_2 \leq z \leq x_{n-3}$, modified fifth degree polynomial interpolation is used, and for $z < x_2$ or $z > x_{n-3}$, quadratic interpolation is used.

SPEED $\sim$0.084 ms (6600, April 1969).

AUTHOR J. Nicholls DATE April 1969

TITLE Cubic Lagrange Interpolation

(ALAGIN uses cubic interpolation. If there is any doubt if this is appropriate, use AIKINT instead.)

ENTRY R=ALAGIN(Z,X,Y,N)

Z real input variable, the value of z for which f(z) is to be calculated.

X,Y real input arrays specifying a table of function values as follows: $Y(i) = f(x_i)$, i=1,...,n. X must be monotonically increasing.

N integer input variable, the number of elements in X and Y.

EXIT This function returns the real value of f(z) to the calling routine. No error checking is performed.

SPEED .168 ms (6600, August 1969).

AUTHOR C. Foxworthy DATE August 1969

TITLE Aitken Interpolation

ENTRY R=AIKINT(Z,X,Y,N,EPS)

Z real input variable, the value of x for which f(x) is to be calculated.

X,Y real input arrays specifying a table of function values as follows: $Y(I) = f(x_i)$, i=1,...,n. X must be monotonically increasing.

N integer input variable, the number of elements in X and Y.

EPS real input variable, the absolute error for convergence, i.e. two successive approximations of the required value must differ in magnitude by less than EPS.

EXIT This function returns the real value of f(z) unless the following errors occur:

If $N \leq 2$, the function exits with an error message and the function value is 0.0, $f(x_1)$, or $f_{1,2}(z)$ (the linearly interpolated value) for $N \leq 0$, N=1, or N=2 respectively.

If convergence is not achieved after ten iterations, or after all the N values in the array have been used if N<10, the routine exits with an error message, stating the value of Z. The function value is the last one evaluated before the differences between successive approximations began to increase.

COMMON BLOCKS USED

AELERCM

ACCURACY Depends on the accuracy requested and the distribution of the given values of X.

SPEED Depends on the accuracy required, the distributions of the arrays and the length of the arrays.

When n=12, it takes approximately 42 μs to obtain a value for AIKINT when three iterations are required. (6600, Sept. 1970)

AUTHOR J.L. Barton DATE September 1970

TITLE Multivariate Interpolation of a Function $f(x_1,x_2,\ldots,x_d)$ where $d \leq 5$.

ENTRY CALL INPOL(T,NT,Z,N,<u>RES</u>)

T real input array to supply the following data:

T(1) contains d, the number of independent variables, $d \leq 5$.

T(2),...,T(d+1) contain $m_1,\ldots,m_d$, the number of values of the independent variable $x_1,\ldots,x_d$, respectively.

The next m_1 cells of T contain the values of x_1, followed by m_2 values of x_2, etc. The values of each independent variable must be distinct and monotonically increasing.

Next come $m_1 m_2 \ldots m_d$ values of the function $f(x_1,\ldots,x_d)$ defined on a rectangular grid. These must be arranged in the form of m_d sets of m_{d-1} set of $\ldots m_1$ values of $f(x_1,\ldots,d_d)$.

NT integer input variable, number of elements in T.

Z real input array of length d specifying the values of the independent variables $z_1,\ldots,z_d$ at which interpolation is required.

N integer input array of length 5 to specify the number of points to be used for interpolation in each dimension. n_i, the number of points to be used for interpolation in the ith dimension, must be ≥ 2 and $\leq \min(30, m_i)$. If n_i is outside these limits, it is set to be the nearest limit. Also,

$$\prod_{i=1}^{d} n_i \text{ must be } < 1000.$$

RES real, output variable, to return the result of the interpolation.

EXIT

This routine exits after storing the result in RES, unless one of the following errors occurs:

(a) $d<1$. Error message printed is INPOL ERROR1.

(b) One of the given z_i is outside the range $(x_i(1),x_i(m_i))$. Error message printed is INPOL ERROR2.

In either case, the job is terminated.

STORAGE

2200_8

ACCURACY

The accuracy depends on the function $f(x_1,...,x_d)$ and the number of points used for interpolation.

SPEED

T(1)	N(I)	TIME (6600, April 1969)
1	8	2 ms
2	4,5	4 ms
5	4,4,5,5,2	62 ms

AUTHOR T. Tan DATE April 1969

1-15 DIFFERENTIATION

00-09 DERIVATIVES OF A TABULATED FUNCTION

1-15-00	DUDX	DUDX1 FIRST DERIVATIVES OF A TABULATED FUNC
1-15-00	DUDXX	DUDXX1 SECOND DERIVATIVES OF A TABULATED FUNC

TITLE

Derivatives du/dx or d^2u/dx^2 of a Tabulated Function, u(x).

For meaningful results the tabulated values should be smooth continuous functions. Experimental data should be smoothed, or the routine HDERIV used instead.

ENTRIES

CALL DUDX(U,DU,X,N,NCUPL) to calculate $\left.\frac{du}{dx}\right|_{x_i}$ i=1,...,n

R = DUDX1(U,X1,X,N,NCUPL) to calculate $\left.\frac{du}{dx}\right|_{x_1}$

CALL DUDXX(U,DU,X,N,NCUPL) to calculate $\left.\frac{d^2u}{dx^2}\right|_{x_i}$ i=1,...,n

R = DUDXX1(U,X1,X,N,NCUPL) to calculate $\left.\frac{d^2u}{dx^2}\right|_{x_1}$

U real input array containing values of the function u_i, i=1,...,n.

DU real output array to return the derivatives du/dx or d^2u/dx^2.

X real input array containing the values of independent variable, X. (X should be monotonically increasing.)

X1 real input variable supplying the point, X1, at which derivatives are required.

N integer input variable, the number of entries in U, DU, X.

NCUPL integer input variable, the number of points to be used in evaluating du/dx or d^2u/dx^2, NCUPL $\leq$ N. The order of the formulae used will be (NCUPL-1).

For all odd NCUPL $\geq$ 3, central differences are used.

For all even NCUPL $\geq$ 4, central differences with one point backward bias are used.

For NCUPL = 1, two point forward differences are used for du/dx.

For NCUPL = 2, two point backward differences are used for du/dx.

For NCUP = 1 and 2 corresponding three point differences are used for d^2u/dx^2.

EXIT The subroutines DUDX and DUDXX return a full array of derivatives in DU. The value of the functions DUDX1 and DUDXX1 are the single derivatives required. No error checking is done.

AUTHOR M.B. Carver DATE August 1974

1-16 INTEGRATION OF USER SUPPLIED FUNCTIONS

00-09	INTEGRATION OF GENERAL MATHEMATICAL FUNCTIONS		
	1-16-00	CADRE	EXTRAPOLATIVE ROMBERG INTEGRATION
	1-16-01	COSIMP	ADAPTIVE SIMPSONS INTEGRATION
	1-16-02	AGAUSS	ADAPTIVE GAUSS INTEGRATION
	1-16-03	QUAD	ADAPTIVE NEWTON-COTES INTEGRATION
10-19	INTEGRATION OF SPECIAL MATHEMATICAL FUNCTIONS		
	1-16-10	FILON	FILON INTEGRATION
	1-16-11	GALA	GAUSS-LAGUERRE INTEGRATION
	1-16-12	GAHER	GAUSS-HERMITE INTEGRATION
	1-16-13	GAJAC	GAUSS-JACOBI INTEGRATION
20-29	MULTIPLE INTEGRATION		
	1-16-20	DUBLINT	GAUSSIAN DOUBLE INTEGRATION
30-39	INTEGRATION OF TABULAR FUNCTIONS		
	1-16-30	SPLINT	INTEGRATION OF TABULAR DATA

1-16 INTEGRATION OF USER SUPPLIED FUNCTIONS

The content of this section of AELIB is the result of an extensive study and evaluation of available quadrature algorithms. The report describing this project [1] is recommended reading for all users doing numerical quadrature. (Note that the writeups for the quadrature routines in this report should be used in preference to those in the appendix of [1].)

In general, all the quadrature routines contain an auxiliary routine to be supplied by the user, and an error flag. The auxiliary routine defines the function, and may call library or other user supplied routines to do so. The error flag is used on input to direct whether messages will be printed, and on output to diagnose that an error has occurred and which error has occurred. Most routines also have as input parameters, the required tolerance or formula order. In all cases, except for the routines dealing with infinite limits, the limits are parameters and must not be infinite or indefinite. The order $A < B$ is implied but not mandatory [2].

Recommended Decision Tree for Selecting a Library Routine [3]

The decision tree on the next page is an attempt to guide the selection of one of the available library routines for a given quadrature problem. For each decision question, a yes indicates proceed to the right, a no indicates proceed downward.

M.B. Carver
June 1978

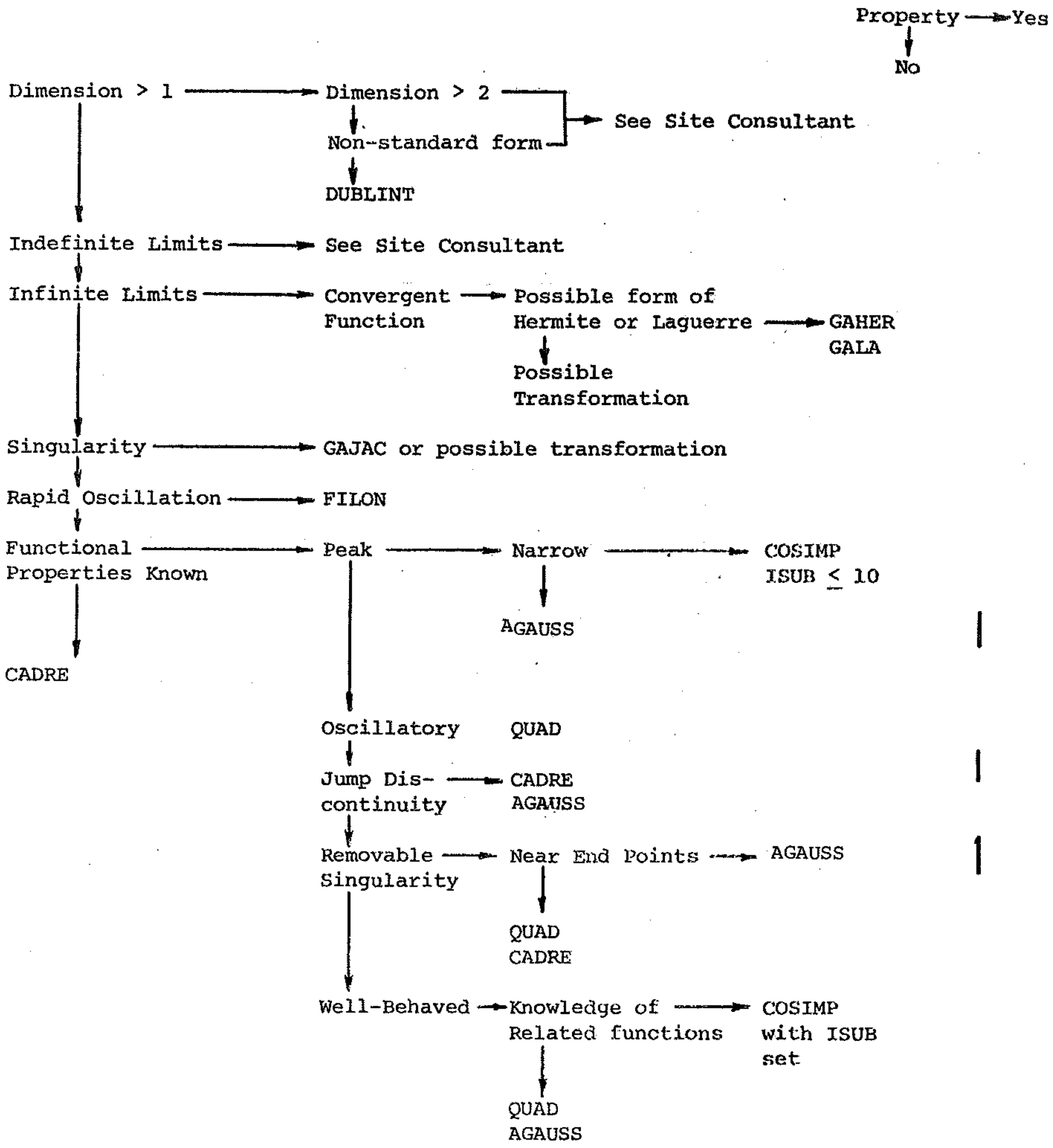
REFERENCES

[1] An Evaluation of Available Quadrature Algorithms and Selection for the AECL FORTRAN Mathematical Library, M.B. Carver and V.J. Jones, May 1977, AECL-5605.

[2] Ibid, pg. 64.

[3] Ibid, pp. 59-60.

Decision Tree for Selecting
a Suitable Quadrature Algorithm(3)



Note: For speed alone where only a general knowledge of accuracy is required, use COSIMP with a suitable ISUB and request a large error tolerance.

TITLE

Integration by Extrapolative Romberg Method

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

INTRODUCTION

Although a large routine, CADRE is probably the best quadrature routine for general use, and is particularly useful for difficult integrands, as it will recognize problem areas and has an option permitting these to be reported in some detail.

ENTRY

R = CADRE(A,B,AUX,EPS,IFLAG)

A,B real input variables, the lower and upper integration limits.

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX." AUX has one parameter, x.

EPS real input parameter to specify the required relative error tolerance in the result. CADRE will subdivide the interval until the error estimated is less than EPS.

IFLAG integer parameter used as input to control the type of diagnostic printout during the progress of integration and as output to return error status as described in EXIT.

On input, if $|\text{IFLAG}|$ is

0 or 1 No printout.

2 List any singularities encountered.

3 In addition, list all subdivisions used to evaluate R, and any regular behaviour associated with each.

4 In addition, list decision procedure to set up internal subdivision.

5 In addition, list the Romberg T tables used (see reference).

EXIT

CADRE returns the estimate of R. This is accurate to a probable tolerance of EPS unless any problems are encountered.

These are reported via IFLAG as described below but all errors are non-fatal.

If IFLAG returns

≥-1 All is well.

-2 One or more singularities were successfully detected and handled.

-3 In some interval(s) the estimate was accepted because the error was small although no predictable behaviour was recognized.

-4 Failure to converge to EPS due to available storage limit being reached by too many function evaluations being required in the interval. The message

***TOO MANY FUNCTION EVALUATIONS REQUIRED IN INTERVAL (_,__) is printed with traceback in this case.

-5 Failure because the interval has become too small in comparison with attainable accuracy. The message ***TOO SMALL A SUBINTERVAL REQUIRED IN INTERVAL (_,__) is printed in this case.

ROUTINES CALLED

AUX, supplied by the user.

COMMON BLOCKS USED:

STATS, PARAM and AELERCM

The common block STATS may be used in the calling program to access the following information on return from CADRE.

COMMON/STATS/EREST,KOUNT,INTS

EREST — real variable, estimate of probable maximum error in the answer obtained as the sum of the errors on accepted intervals.

KOUNT — integer variable, number of function evaluations performed to compute R to within EPS.

INTS — integer variable, the number of subdivisions needed to compute R within EPS.

ACCURACY

Should be within the relative tolerance of EPS if IFLAG is ≥ -3.

STORAGE

6600_8

EXAMPLE

The following code uses CADRE to evaluate

$$R = \int_0^2 f(x)dx$$

$$\text{for } f(x) = \begin{cases} \sqrt{x}, & 0 \leq x \leq 1 \\ x+2, & 1 < x \leq 2 \end{cases}$$

CADRE diagnoses the discontinuity at x=1 and the characteristic behaviour of $\sqrt{x}$.

```
      PROGRAM TCADRE(OUTPUT)
C
      EXTERNAL DISC
C
      PRINT *,"    TEST AND ILLUSTRATE CADRE          "
C
      EXACT=3.5+2./3.
      A=Ø
      B=2.
      IFLAG=2
      EPS=1.ØE-5
C
      RINT=CADRE(A,B,DISC,EPS,IFLAG)
C
      PRINT *,"    EXACT ANSWER                     ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION          ",RINT
      PRINT *,"    ERROR                            ",ABS(EXACT-RINT)
      PRINT *,"    IFLAG RETURNS                    ",IFLAG
C
      STOP
      END

      FUNCTION DISC(X)
      DISC=SQRT(X)
      IF(X.LE.1)RETURN
      DISC=X+2
      RETURN
      END
```

```
   TEST AND ILLUSTRATE CADRE
      INTEGRAND SHOWS SINGULAR BEHAVIOR OF TYPE X**( .5Ø)
BETWEEN  Ø.              AND    .1ØØØØØØØE+Ø1
       INTEGRAND SEEMS TO HAVE JUMP OF SIZE   .2ØØØØØE+Ø1
BETWEEN  .1ØØØØØØØE+Ø1 AND    .1ØØØØ153E+Ø1
   EXACT ANSWER                   4.166666666667
   RESULT FROM INTEGRATION        4.166662884316
   ERROR                          .ØØØØØ3782350688653
   IFLAG RETURNS                  -2
```

SOURCE C. DeBoor, "An Algorithm for Numerical Quadrature", Mathematical Software, J.R. Rice, Editor, Academic Press, 1971.

ADAPTED FOR AELIB by Valerie Jones DATE August 1976

REFERENCE M.B. Carver and V.J. Jones, AECL-5605.

TITLE

Integration by Adaptive Simpson's Rule

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

INTRODUCTION

COSIMP is a small fast quadrature routine for well-behaved integrands and is particularly useful for integrating several related integrands, or integrands with narrow peaks, as the amount of subdivision of the integral can be specified.

ENTRY

R = COSIMP(A,B,AUX,EPS,ISUB)

A,B real input variables, the lower and upper integration limits.

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX." AUX has one parameter, x.

EPS real input parameter to specify the required relative error tolerance. COSIMP will subdivide the interval until the error estimated is less than EPS.

ISUB Integer parameter used as input to control the initial subdivision and diagnostic printout and as output to return error status. The interval is estimated initially using

$$N1 = 2^{|ISUB|} \text{ and } N2 = 2N1$$

subdivisions. If these agree within a relative error of EPS, the second result is returned. Otherwise, the intervals are successively halved until successive estimates converge to within EPS. |ISUB| must be less than 19 as a maximum of 2^{20} divisions are allowed. ISUB may be set negative to activate error diagnostics. It returns the required subdivision, as described in EXIT.

EXIT

COSIMP returns the estimate of R, which is within a probable tolerance of EPS if converged. ISUB is returned such that the number of subdivisions used was

$2^{|\text{ISUB}|}$.

When the calculation converges, ISUB will be returned positive, but if convergence was not attained with 2^{20} subdivisions, and if activated by entering a negative ISUB, the following message is printed with tracework:

***TROUBLE INTEGRATING OVER (__,__). EST. INTEGRAL AND ERROR ARE __ AND __.

The estimate of the interval is accepted and integration continues. ISUB is then returned as -20. If higher accuracy is required, the integral can later be evaluated separately over the problem interval.

ROUTINES CALLED

AUX, supplied by the user.

COMMON BLOCKS USED

QSTATS and AELERCM.

The common block QSTATS defined by

```
COMMON/QSTATS/EREST,KOUNT,INTS
```

may be used in the same way as the common block STATS in CADRE to get further information from COSIMP.

ACCURACY

Should be within a relative tolerance of EPS unless ISUB returns -20.

EXAMPLE

The following code uses COSIMP to evaluate

$$R = \int_0^1 \frac{dx}{1+e^x} .$$

The results are shown.

```
      PROGRAM TCOSIMP(OUTPUT)
C
      EXTERNAL COSF
C
      PRINT *,"    TEST AND ILLUSTRATE COSIMP       "
C
      EXACT=.3798854930417
      A=Ø
      B=1
      EPS=1.ØE-5
      IFLAG=Ø
C
      RINT=COSIMP(A,B,COSF,EPS,IFLAG)
C
      PRINT *,"    EXACT ANSWER                    ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION         ",RINT
      PRINT *,"    ERROR                           ",ABS(EXACT-RINT)
      PRINT *,"    IFLAG RETURNS                   ",IFLAG
C
      STOP
      END


      FUNCTION COSF(X)
      COSF=1/(1+EXP(X))
      RETURN
      END
```

```
      TEST AND ILLUSTRATE COSIMP
      EXACT ANSWER                   .3798854930417
      RESULT FROM INTEGRATION        .3798834965601
      ERROR                          .0000019964815 7384
      IFLAG RETURNS                  2
```

PROGRAMMED BY D.H. Howden in APEX Language, 1965.
C.J. Johnson in FORTRAN, 1969.

REVISED BY Valerie Jones, August 1976.

REFERENCE M.B. Carver and V.J. Jones, AECL-5605.

TITLE

Integration by Adaptive Gauss Method.

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

INTRODUCTION

AGAUSS is a small routine which does not store any intermediate function values. Because of this it can divide intervals very finely if necessary, but is slower than similar routines. It is useful for integrals containing peaks, singularities or discontinuities. It can use up to 2^{50} subintervals.

ENTRY

R = AGAUSS(A,B,AUX,EPS,IFLAG)

A,B real input variables, the lower and upper integration limits.

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX." AUX has one parameter, x.

EPS real input parameter to specify the required relative error tolerance in the result. AGAUSS will subdivide the interval until the error estimate is less than EPS.

IFLAG integer parameter used as input to control diagnostic printout during the progress of integration as described below and as output to report error status as described in EXIT.

If input as

0 No printout.

≠0 List any problems encountered.

EXIT

AGAUSS returns the estimate of R. This is accurate to a probable tolerance of EPS, unless any problems are encountered. These are reported via IFLAG as follows:

If IFLAG returns

0   All is well.

<0 |IFLAG| integration problems were detected.

In the latter case, if input value of IFLAG was ≠0, the following message is printed with tracework for each problem area.

***TROUBLE INTEGRATING OVER (__,__). EST. INTEGRAL AND ERROR ARE __ AND __ .

The estimate of the interval is accepted and integration continues. IFLAG is then decremented by 1. If higher accuracy is required, the integral can later be evaluated separately over the problem interval.

ROUTINES CALLED   AUX, supplied by the user.

COMMON BLOCKS USED   QSTATS and AELERCM

The common block QSTATS defined by

```
   COMMON/QSTATS/EREST,KOUNT,INTS
```

may be used in the same way as the common block STATS in CADRE to get further information for AGAUSS.

ACCURACY   Should be within a relative tolerance of EPS unless IFLAG returns negative.

EXAMPLE   The following code uses AGAUSS to evaluate

$$\int_0^1 \frac{x\,dx}{e^x-1}$$

The results are shown.

```
      PROGRAM TAGAUSS(OUTPUT)
      COMMON/QSTATS/EREST,KOUNT,INTS
C
      EXTERNAL EFUN
C
      PRINT *,"    TEST AND ILLUSTRATE AGAUSS          "
C
      EXACT=.7775Ø463411
      A=Ø
      B=1
      EPS=1.ØE-5
C
      RINT=AGAUSS(A,B,EFUN,EPS,IFLAG)
C
      PRINT *,"    EXACT ANSWER                  ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION       ",RINT
      PRINT *,"    ERROR                         ",ABS(EXACT-RINT)
      PRINT *,"    ESTIMATED ERROR               ",EREST
      PRINT *,"    NO OF FUNCTION EVALUATIONS    ",KOUNT
      PRINT *,"    NO OF SUBDIVISIONS            ",INTS
      PRINT *,"    IFLAG RETURNS                 ",IFLAG
C
      STOP
      END

      FUNCTION EFUN(X)
C
C     DEFINE THE FUNCTION . AS AGAUSS DOES NOT EXPLICITLY USE ENDPOINTS
C               DO NOT WORRY ABOUT THE SINGULARITY AT X=Ø.
C
      EFUN=X/(EXP(X)-1)
      RETURN
      END
```

```
      TEST AND ILLUSTRATE AGAUSS
      EXACT ANSWER                  .7775Ø463411
      RESULT FROM INTEGRATION       .7775Ø46341123
      ERROR                         2.3Ø5711177542E-12
      ESTIMATED ERROR               1.Ø6581410364E-13
      NO OF FUNCTION EVALUATIONS    11
      NO OF SUBDIVISIONS            1
      IFLAG RETURNS                 Ø
```

SOURCE D.K. Kahanar, "Comparison of Numerical Quadrature Formulas", Mathematical Software, J.R. Rice, Editor, Academic Press, 1971.

ADAPTED FOR AELIB BY Valerie Jones DATE August 1976

REFERENCE M.B. Carver and V.J. Jones, AECL-5605.

TITLE

Integration by Adaptive Newton Cotes Method.

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

INTRODUCTION

QUAD is an efficient routine which stores function evaluations to attain greater speed. It is particularly good for oscillatory functions and is reliable at detecting peaks and singularities but is sometimes unable to subdivide the interval sufficiently to integrate over these. It can use a maximum of 2^{20} intervals.

ENTRY

R = QUAD(A,B,AUX,EPS,IFLAG)

A,B real input variables, the lower and upper integration limits.

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX." AUX has one parameter, x.

EPS real input parameter to specify the required relative error tolerance in the result.

IFLAG integer parameter used as input to control diagnostic printout during the progress of integration as described below and as output to return error status as described in EXIT.

If input as

0	No printout.
≠0	List any problems encountered.

EXIT

QUAD returns the estimate of R. This should be accurate to a probable tolerance of EPS unless any problems are encountered. These are reported via IFLAG as follows:

If IFLAG returns

0 All is well.

<0 |IFLAG| integration problems were detected.

In the latter case, if input value of IFLAG was ≠0, the following message is printed with traceback for each problem area:

```
***TROUBLE INTEGRATING OVER (__,__).  EST. INTEGRAL
AND ERROR ARE __ AND __.
```

The estimate of the interval is accepted and integration continues. If higher accuracy is required, the integral can later be evaluated separately over the problem interval.

ROUTINES CALLED AUX, supplied by the user

COMMON BLOCKS USED QSTATS and AELERCM.

The common block QSTATS defined by

```
COMMON/QSTATS/EREST,KOUNT,INTS
```

may be used in the same way as the common block STATS in CADRE to yet further information from QUAD.

ACCURACY Should be within relative error of EPS unless IFLAG returns negative.

STORAGE $\sim 1350_8$

EXAMPLE The following code uses QUAD to evaluate

$\int_0^1 \frac{2dx}{2+\sin(10\pi x)}$. The results are shown.

```
      PROGRAM TQUAD(OUTPUT)
C
      EXTERNAL SINFUN
C
      PRINT *,"    TEST AND ILLUSTRATE QUAD           "
C
      EXACT=1.15470066904
      A=0
      B=1
      IFLAG=1
      EPS=1.0E-5
C
      RINT=QUAD(A,B,SINFUN,EPS,IFLAG)
C
      PRINT *,"    EXACT ANSWER                       ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION            ",RINT
      PRINT *,"    ERROR                              ",ABS(EXACT-RINT)
      PRINT *,"    NUMBER OF POINTS REQUIRED          ",N
C
      STOP
      END


      FUNCTION SINFUN(X)
      DATA PI /3.1415926 /
      SINFUN=2/(2+SIN(10*PI*X))
      RETURN
      END
```

```
TEST AND ILLUSTRATE QUAD
EXACT ANSWER                 1.15470066904
RESULT FROM INTEGRATION      1.154700600659
ERROR                        6.838124733122E-8
NUMBER OF POINTS REQUIRED    0
```

SOURCE D.K. Kahaner, "Comparison of Numerical Quadrature Formulas", Mathematical Software, J.R. Rice, Editor, Academic Press, 1971.

ADAPTED FOR AELIB BY

M.B. Carver and V.J. Jones, AECL-5605.

TITLE Filon Integration

$$c = \int_a^b f(x)\cos(wx)dx \quad \text{or} \quad s = \int_a^b f(x)\sin(wx)dx$$

INTRODUCTION FILON is designed for integrands with oscillating components.

ENTRY CALL FILON(AUX,W,A,B,EPS,M,C,NC,S,NS,IFLAG)

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX." AUX has one parameter, x.

W real input parameter containing the frequency component.

A,B real input variables, the limits of the integration.

EPS real input parameter to specify the required relative error tolerance in the result.

M integer input variable, the maximum number of subdivisions to be considered is 2^M. The interval (A,B) is subdivided successively, until two successive estimates converge to within EPS.

C,S real output parameters to receive integrals as defined above.

NC,NS integer input parameters. If NC=1 at entry, the routine computes C. NS similarly controls S. On exit NC and NS report required subdivision.

IFLAG integer parameter used as input to control error printing and as output to report errors as described in EXIT.

EXIT If convergence is achieved, C, S return the integrals and the number of required subdivisions for each is 2^{NC} and 2^{NS}. IFLAG is unchanged.

If convergence is not achieved, C, S return the estimate for 2^M subdivisions, the offender of NC and/or NS is set to -M, and IFLAG is set negative. Unless IFLAG was set zero on entry to suppress printing, the message
***FILON FAILED TO CONVERGE TO REQUIRED TOLERANCE IN 2__
SUB DIVISIONS
is printed with traceback where __ will be the value of M.

ROUTINES CALLED
AUX, supplied by the user.

COMMON BLOCKS USED
AELERCM

ACCURACY
Should be within EPS if IFLAG is not returned negative.

EXAMPLE
The following shows the code to estimate

$$\int_0^\pi e^{4x} \sin(2.5x)dx \text{ and } \int_0^\pi e^{4x} \cos(2.5x)dx$$

using FILON, and the results obtained.

```
      PROGRAM TFILON(OUTPUT)
C
      EXTERNAL FILF
C
      PRINT *,"    TEST AND ILLUSTRATE FILON         "
C
      EXAST=-6Ø8.78717Ø58639
      EXACT=-169.345717190
      A=Ø
      B=2
      W=2.5
      NC=NS=1
      M=2Ø
      EPS=1.ØE-5
C
      CALL FILON(FILF,W,A,B,EPS,M,C,NC,S,NS,IFLAG)
C
      PRINT *,"    EXACT VALUES                     ",EXACT,EXAST
      PRINT *,"    RESULTS FROM INTEGRATION         ",C,S
      PRINT *,"    ERRORS                          ",ABS(EXACT-C),
                                                     ABS(EXAST-S)
     'PRINT *,"    NO OF SUBDIVISIONS       2**    ",NC,NS
      PRINT *,"    IFLAG RETURNS                   ",IFLAG
C
      STOP
      END


       FUNCTION FILF(X)
       FILF=EXP(4*X)
       RETURN
       END
```

```
TEST AND ILLUSTRATE FILON
EXACT VALUES                     -169.34571719 -6Ø8.7871Ø5864
RESULTS FROM INTEGRATION         -169.34571Ø1Ø69 -6Ø8.7872581403
ERRORS                          .ØØØØØ7Ø83Ø938Ø57Ø4 .ØØØØ8755391172599
NO OF SUBDIVISIONS       2**    8 7
IFLAG RETURNS                   Ø
```

SOURCE   Chase and Fosdick, "Algorithm for Filon Quadrature",
              Comm. ACM, Vol. 12, 1969.

PROGRAMMED BY M.B. Carver   July 1970

REVISED BY   M.B. Carver   August 1976

REFERENCE   M.B. Carver and V.J. Jones, AECL-5605.

TITLE

Gauss-Laguerre Integration

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

INTRODUCTION

Because of the time involved in calculation, and the amount of space required to store the constants needed for Gauss-Laguerre Integration, this routine is limited to formulas involving 4, 8, 16, 32 or 64-points. This provides adequate accuracy for integrals where f(x) is well-behaved, but may be inadequate for complicated functions f(x).

ENTRY

R = GALA(AUX,N,EPS)

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX". AUX has one argument, x.

N integer parameter used
(a) as input to specify to the number of points to use, or
(b) as input to control error reporting and as output to return error status as described in EXIT.

EPS real input parameter to specify required relative error tolerance in the result, see EXIT.

EXIT

(a) If N is input as one of 4, 8, 16, 32, 64, the integral is evaluated only once using an N point formula, EPS is not used, there is no estimation of accuracy, and N is returned unchanged.

(b) If N is set to any other integer, the routine begins by evaluating A for N=4 and N=8(EST8). If the difference between these two estimates is less than EPS*EST8, the estimate with N=8 is returned. If not, the routine increases N through 16, 32, and 64 until two successive estimates converge. N is returned containing the current value to permit this to be used for subsequent calls for related functions.

If converge has not been attained when N reaches 64, the last estimate is returned, N is set to -64, and unless N was input as zero to suppress printing, the message

***REQUESTED ERROR NOT ATTAINED. ERROR IN GALA PROBABLY LESS THAN ___

is printed with traceback where ___ is the difference between the last two estimates.

COMMON BLOCKS USED AELERCM

ROUTINES CALLED AUX, supplied by the user.

ACCURACY Unknown for case (a) unless approximately known from related functions.

For case (b) the result should be accurate to EPS if N is returned positive.

STORAGE 750_8

EXAMPLE The following shows code to estimate

$$R = \int_0^\infty e^{-x} x^{10} dx$$

using GALA, and the results obtained.

```
      PROGRAM TGALA(OUTPUT)
C
      EXTERNAL X1Ø
C
      PRINT *,"    TEST AND ILLUSTRATE GALA          "
C
      EXACT=3.6288EØ6
      A=Ø
      B=1
      N=1
      EPS=1.ØE-5
C
      RINT=GALA(X1Ø,N,EPS)
C
      PRINT *,"    EXACT ANSWER                      ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION           ",RINT
      PRINT *,"    ERROR                             ",ABS(EXACT-RINT)
      PRINT *,"    NUMBER OF POINTS REQUIRED         ",N
C
      STOP
      END


       FUNCTION X1Ø(X)
       X1Ø=X**1Ø
       RETURN
       END
```

```
TEST AND ILLUSTRATE GALA
EXACT ANSWER                  36288ØØ.
RESULT FROM INTEGRATION       36288ØØ.
ERROR                         1.639127731323E-7
NUMBER OF POINTS REQUIRED     16
```

SOURCE Stroud and Secrest, "Gaussian Quadrature Formulae", Prentice Hall, 1966.

PROGRAMMED BY Judy Barton, April 1971.

REVISED BY Valerie Jones, August 1976.

REFERENCE M.B. Carver and V.J. Jones, AECL-5605

TITLE

Gauss-Hermite Integration

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

INTRODUCTION

Because of the time involved in calculation, and the amount of space required to store the constants needed for Gauss-Hermite Integration, this routine is limited to formulas involving 4, 8, 16, 32 or 64-points. This provides adequate accuracy for integrals where f(x) is well-behaved, but may be inadequate for complicated functions f(x).

ENTRY

R = GAHER(AUX,N,EPS)

AUX real function subprogram supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX". AUX has one argument, x.

N integer parameter used
(a) as input to specify the number of points to use, or
(b) as input to cpmtrp; error reporting and as output to return error status as described in EXIT.

EPS real input parameter to specify required relative error tolerance in the result, see EXIT.

EXIT

(a) If N is input as one of 4, 8, 16, 32, 64, the integral is evaluated only once using an N point formula, EPS is not used and there is no estimation of accuracy.

(b) If N is set to any other integer, the routine begins by evaluating A for N=4 and N=8 (EST8). If the difference between these two estimates is less than EPS*EST8, the estimate with N=8 is returned. If not, the routine increases N through 16, 32, and 64 until two successive estimates converge. N is returned containing the current value to permit this to be used for subsequent calls for related functions.

If convergence has not been attained when N reaches 64, the last estimate is returned, N is set to -64, and unless N was input as zero to suppress printing, the message

```
***REQUESTED ERROR NOT ATTAINED.   ERROR IN GAHER
PROBABLY LESS THAN ____
```

is printed with traceback, where ____ is the difference between the last two estimates.

COMMON BLOCKS USED

AELERCM

ROUTINES CALLED

AUX, supplied by the user.

ACCURACY

Unknown for case (a) unless approximately known from related functions.

For case (b) the result should be accurate to EPS if N is returned positive.

EXAMPLE

The following shows code to estimate

$$R = \int_{-\infty}^{\infty} e^{-x^2} \cos(3x)dx$$

using GAHER, and the results obtained.

```
      PROGRAM TGAHER(OUTPUT)
C
      EXTERNAL FCOS
C
      PRINT *,"    TEST AND ILLUSTRATE GAHER        "
C
      EXACT=SQRT(XPI(1.))*EXP(-9./4.)
      A=Ø
      B=1
      N=1
      EPS=1.ØE-5
C
 1ØØ  RINT=GAHER(FCOS,N,EPS)
C
      PRINT *,"    EXACT ANSWER                     ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION          ",RINT
      PRINT *,"    ERROR                            ",ABS(EXACT-RINT)
      IF(EPS.EQ.Ø)STOP
      PRINT *,"    NUMBER OF POINTS REQUIRED        ",N
C
      PRINT *,"    REPEAT FOR FIXED N (=8) "
      EPS=Ø
      N=8
      GOTO 1ØØ
C
      END


      FUNCTION FCOS(X)
      FCOS=COS(3*X)
      RETURN
      END
```

```
TEST AND ILLUSTRATE GAHER
EXACT ANSWER                   .1868152614571
RESULT FROM INTEGRATION        .1868152614571
ERROR                          1.33226762955E-14
NUMBER OF POINTS REQUIRED      32
REPEAT FOR FIXED N (=8)
EXACT ANSWER                   .1868152614571
RESULT FROM INTEGRATION        .1866187196885
ERROR                          .ØØØ1965417686423
```

SOURCE Stroud and Secrest, "Gaussian Quadrature Formulae", Prentice Hall, 1966.

PROGRAMMED BY Judy Barton, April 1971.

REVISED BY Valerie Jones, August 1976.

REFERENCE M.B. Carver and V.J. Jones, AECL-5605.

TITLE Gauss-Jacobi Integration

$$R = \int_a^b (b-x)^{\alpha}(x-a)^{\beta} f(x)dx$$

INTRODUCTION GAJAC is designed to integrate functions which can be written in the form $g(x) = (b-x)^{\alpha}(x-a)^{\beta}\ f(x)$, which contain a singularity at one or both end points if α and/or β are negative. As the nodes and weights of the corresponding n point Jacobi polynomial depend on α and β they must be calculated at least once for each set of (a,b,α,β,n). This is done by an internal call to the associated routine JACOBI.

ENTRY R = GAJAC(AUX,A,B,ALF,BET,N,EPS)

AUX real function subroutine supplied by the user to evaluate f(x). The calling program must contain the statement "EXTERNAL AUX". AUX has one argument, x.

A,B real input variables containing the integration end points a,b which may also be singularities of g(x).

ALF,BET real input variables containing α,β which if negative cause the singularities.

N integer parameter used

(a) as input to specify the number of points to use and whether the nodes and weights of the Jacobi polynomial have been previously calculated, or
(b) as input to control error reporting and as output to report error status as described in EXIT.

EPS real input parameter to specify required relative error tolerance in the result, see EXIT.

EXIT (a) If N is other than zero or one, the integral is estimated only once using an $|N|$ point formula, EPS is not used, there is no estimation of accuracy and N is returned unchanged. If N is positive, the roots of the Jacobi polynomial are calculated, if negative the roots used by the previous call are used (this assumes that the previous call had the same (a,b,α,β,n) set).

(b) If N is zero or one, the integral is estimated for N=4 (EST4) and 8 points (EST8). If the difference between these two estimates is less than EPS*EST8, EST8 is returned. Otherwise, the routine increases N through 16, 32, 64, 128 points until two successive estimates converge to within EPS*the most recent estimate. N is returned at the current value to permit this to be used for subsequent calls to related functions.

If convergence is still not attained when N reaches 128, the last estimate is returned, N is set to - -128, and unless N was input as zero to suppress printing, the message

***REQUESTED ERROR NOT ATTAINED, ABSOLUTE ERROR IN GAJAC PROBABLY LESS THAN ___

is printed with traceback, where ___ is the difference between the last two estimates.

COMMON BLOCKS USED AELERCM

ROUTINES CALLED JACOBI, a utility routine loaded with GAJAC and AUX, supplied by the user

ACCURACY Unknown for case (a) unless approximately known for related functions.

For case (b) the results should be accurate to EPS if N is returned positive.

STORAGE 1720_8 GAJAC 1310_8
JACOBI 410_8

EXAMPLE

The following program estimates the integral

$$R = \int_0^1 |x-1/3|^{-.7}dx$$

by splitting it into two, such that the singularity now resides at the end points

$$R = \int_0^{1/3}(1/3-x)^{-.7}(x-0)^0.1.dx + \int_{1/3}^1(1-x)^0(x-1/3)^{-.7}.1.dx$$

The required number of points is established for the first integral and the same number used for the second.

```
      PROGRAM TGAJAC(OUTPUT)
C
      EXTERNAL FMOD
C
      PRINT *,"    TEST AND ILLUSTRATE GAJAC          "
C
C
      EXACT=5.348968622
      ALF=-.7
      BET=Ø
      THIRD=1./3.
      N=1
      A=Ø
      B=1
      EPS=1.ØE-5
C
      RINT=GAJAC(FMOD,A,THIRD,ALF,BET,N,EPS)
     ,    +GAJAC(FMOD,THIRD,B,BET,ALF,N,EPS)
C
      PRINT *,"    EXACT ANSWER                ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION     ",RINT
      PRINT *,"    ERROR                       ",ABS(EXACT-RINT)
      PRINT *,"    NUMBER OF POINTS REQUIRED   ",N
C
      STOP
      END


      FUNCTION FMOD(X)
      FMOD=1.
      RETURN
      END
```

```
TEST AND ILLUSTRATE GAJAC
EXACT ANSWER                  5.348968622
RESULT FROM INTEGRATION       5.348968622055
ERROR                         5.485389920068E-11
NUMBER OF POINTS REQUIRED     8
```

SOURCE Stroud and Secrest, "Gaussian Quadrature Formulae", Prentice Hall, 1966.

PROGRAMMED BY Suzanne Berube, December 1972.

REVISED BY Valerie Jones, August 1976.

REFERENCE M.B. Carver and V.J. Jones, AECL-5605.

TITLE Double Integral Using the Gaussian Method

$$R = \int_a^b g(y)dy \int_{c(y)}^{d(y)} f(x,y)dx$$

ENTRY R = DUBLINT(G,A,B,NO,F,C,D,NI)

G,F,C,D

real function subprograms supplied by the user to do the following:

G calculates g(y)(one formal parameter, y).

F calculates f(x,y)(two formal parameters, x and y).

C calculates the limit c(y)(one formal parameter, y).

D calculates the limit d(y)(one formal parameter, y).

All these functions must exit with the function name as the left-hand identifier of a replacement statement.

The calling program must contain the statement, "EXTERNAL G,F,C,D".

A,B real input variables, the lower and upper limits, respectively, of the outer integral.

NO integer input variable, = 4, 6, 8, 16 or 32, number of points in the integration of the outer integral.

NI integer input variable, = 4, 6, 8, 16 or 32, number of points in the integration of the inner integral.

If NO or NI is not one of the above values, NO or NI = 4 will be assumed and one of the following messages will be printed with traceback to so warn the user:

```
***ERROR IN SPECIFYING NO, NO=4 IS ASSUMED
***ERROR IN SPECIFYING NI, NI=4 IS ASSUMED
```

EXIT — This function returns the value of the double integral.

ROUTINES CALLED — Functions G,F,C, and D supplied by the user.

ACCURACY — Increases with NI and NO but DUBLINT makes no estimate of accuracy.

EXAMPLE — This program uses DUBLINT to find the volume of the unit sphere

$$V = \int_{-1}^{1} 2 \int_{-\sqrt{1-y}^2}^{+\sqrt{1-y}^2} \sqrt{1-x^2-y^2}\ dxdy$$

using 32 data points for both inner and outer integrals. It prints the calculated value and the theoretical value.

```
      PROGRAM TDUBINT(OUTPUT)
C
      EXTERNAL F,G,C,D
C
      PRINT *,"    TEST AND ILLUSTRATE DUBLINT     "
C
C
      EXACT=XPI(4./3.)
      A=-1
      B=1
      NO=1
      EPS=1.ØE-5
      NO=NI=32
C
      RINT=DUBLINT(G,A,B,NO,F,C,D,NI)
C
      PRINT *,"    EXACT ANSWER                  ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION       ",RINT
      PRINT *,"    ERROR                         ",ABS(EXACT-RINT)
C
      STOP
      END
      FUNCTION F(X,Y)
      F=SQRT(1-X**2-Y**2)
      RETURN
      END

      FUNCTION G(Y)
      G=2.
      RETURN
      END

      FUNCTION C(Y)
      C=-SQRT(1-Y**2)
      RETURN
      END

      FUNCTION D(Y)
      D=+SQRT(1-Y**2)
      RETURN
      END
```

```
TEST AND ILLUSTRATE DUBLINT
EXACT ANSWER                 4.188790204786
RESULT FROM INTEGRATION      4.188854244009
ERROR                        .00006403922276377
```

PROGRAMMED BY E.A. Okazaki in APEX Language, November 1962.
T. Tan in FORTRAN, January 1967.

REVISED BY K. Chaplin, May 1976.

REFERENCE M.B. Carver and V.J. Jones, AECL-5605.

TITLE

Integration of Tabular Data with Provision for Subsequent Interpolation.

$\int_a^b y(x)dx$

INTRODUCTION

SPLINT will integrate equally or unequally spaced tabular data and also return parameters which can be used for subsequent interpolation, as it uses a cubic spline fit to the data.

ENTRY

R = SPLINT(X,Y,N,A,B,<u>Y1</u>,<u>Y2</u>,<u>Y3</u>,E,<u>IFLAG</u>)

X real input array dimension N in monotonic ascending sequence containing values of the independent variable.

Y real input array dimension N containing corresponding values of y(x).

N integer input variable containing number of points, which must be $\geq$ 3.

A,B real input variables, the limits of integration, not necessarily coincident with any x.

Y1,Y2,Y3 real output arrays dimension N to return values needed to interpolate using the spline formula

$$y(x) = y(I) + y1(I)*(x-x(I)) + y2(I)*(x-x(I))^2$$
$$+ y3(I)*(x-x(I))^3 , x(I) \leq x \leq x(I+1)$$

E real input array dimension 2N which is needed for workspace.

IFLAG integer parameter to control diagnostic printout and report any errors as described in EXIT.

If IFLAG has input value

0 No printout of errors.

>0 Error messages printed.

EXIT

SPLINT returns the estimate of the integral unless errors are encountered. These are reported via IFLAG as follows. If IFLAG returns

The same value as was input to SPLINT, then all is well.

-1 then N has been detected < 3.

-2 then X has been detected not monotonic.

In either event, integration cannot proceed. SPLINT returns the value 0. If IFLAG has been input non zero, one of the following messages is printed:

```
***N=__ IN SPLINT AT LEAST 3 POINTS REQUIRED
***X___ AND ____ NOT IN ASCENDING ORDER IN SPLINT
```

COMMON BLOCKS USED

AELERCM

ACCURACY

Accuracy is data dependent.

EXAMPLE

The following code uses SPLINT to evaluate the integral from a table which is generated artificially for illustration by evaluating $y(x) = e^x$ at 10 unequally spaced points:

$$R = \int_0^1 y(x)dx$$

and evaluates y(0.5) by the interpolation formula. The results are shown.

```
      PROGRAM TSPLINT(OUTPUT)
C
      REAL X(11),Y(11),Y1(11),Y2(11),Y3(11),E(22)
C
      DATA X/Ø.,.Ø79,.163,.211,.359,.48Ø,.533,.621,.777,.911,1.Ø/
C
C
      PRINT *,"    TEST AND ILLUSTRATE SPLINT      "
C
C
C     SET UP Y ARRAY ARTIFICIALLY USING Y(X)=EXP(X)
      N=11
      DO 2Ø I=1,N
   2Ø Y(I)=EXP(X(I))
C
      EXACT=1.718281828
      A=Ø
      B=1
      EPS=1.ØE-5
C
      RINT=SPLINT(X,Y,N,A,B,Y1,Y2,Y3,E,IFLAG)
C
      PRINT *,"    EXACT ANSWER                       ",EXACT
      PRINT *,"    RESULT FROM INTEGRATION            ",RINT
      PRINT *,"    ERROR                              ",ABS(EXACT-RINT)
C
C         DETERMINE Y AT X=Ø.5
C
C
      Z=Ø.5-X(6)
      YMID=Y(6)+Y1(6)*Z  +Y2(6)*Z**2+Y3(6)*Z**3
      Z=EXP(Ø.5)
C
      PRINT *,"    Y(Ø.5) SHOULD BE                   ",Z
      PRINT *,"    Y(Ø.5) FROM SPLINT                 ",YMID
      PRINT *,"    Y(Ø.5) ERROR                       ",ABS(Z-YMID)
C
C
      STOP
      END
```

```
TEST AND ILLUSTRATE SPLINT
EXACT ANSWER                   1.718281828
RESULT FROM INTEGRATION        1.71832441339
ERROR                          .ØØØØ4258538974256
Y(Ø.5) SHOULD BE               1.6487212707
Y(Ø.5) FROM SPLINT             1.648723144392
Y(Ø.5) ERROR                   .ØØØØØ1873691743981
```

SOURCE   Davis and Rabinowitz, "Methods of Numerical Integration",
              Academic Press, 1975.

REVISED BY   Valerie Jones   DATE   August 1976

REFERENCE   M.B. Carver and V.J. Jones, AECL-5605.

1-17 INTEGRATION OF ORDINARY DIFFERENTIAL EQUATIONS

```
00-09     FIXED STEP , FIXED ORDER METHODS
          1-17-00   RKONE     SHORT RUNGE-KUTTA INTEGRATION
10-19     ERROR CONTROLLED VARIABLE STEP,FIXED ORDER METHODS
          1-17-10   RKFINT    INTEGRATION USING RUNGE-KUTTA-FEHLBERG
20-29     ERROR CONTROLLED VARIABLE STEP,VARIABLE ORDER METH
          1-17-20   STIFFZ    INTEGRATION USING GEARS ALGORITHM
```

1-17 INTEGRATION OF ORDINARY DIFFERENTIAL EQUATIONS

Introduction

Any set of ordinary differential equations can be reduced to the explicit definition of the initial value problem

$$\frac{d}{dt} y_i(t) = f_i(t,y_1,y_2,\ldots,y_i,\ldots,y_n) \quad i=1,\ldots,n$$

$$y_i(0) = y_{oi} \quad i=1,\ldots,n$$

where the y_i are n dependent variables and y_{oi} are the initial conditions on the variables y_i.

This set is normally solved over a prescribed interval in t by setting the initial values $y_i(0)$ and then calling an integration algorithm which completes the integration by repeatedly calling an auxiliary subroutine in which the functions f are defined.

The accuracy of the solution thus obtained naturally depends on the number and size of steps used to cover the integration interval. A good algorithm controls accuracy by adjusting the step size to maintain an error estimate below a given maximum acceptable value.

The difficulty of solving differential equation sets efficiently and accurately, increases directly with the size of the equation set, its non-linearity, and its stiffness, i.e. the spread of magnitude of the time constants of the various transient components. Another difficulty is the presence of discontinuities in the definition of the functions f_i. These topics are reviewed in detail in [1].

AELIB Routines

AELIB has two main integration subroutines:

i) RKFINT - A small, fast routine using the Runge Kutta Fehlberg integration algorithm which uses a combination of a fourth order Runge Kutta method and associated fifth order error estimate to control its step size.

ii) STIFFZ - A large package of routines with the option of using either Gear or Adams predictor corrector formulae of variable orders, and their associated error estimates to control its step size. The efficiency of these methods depends heavily on the accuracy and economy with which the Jacobian matrix elements $J_{ij} = \partial f_i/\partial y_j$ can be computed. Various options for doing this are included in the package.

As RKFINT and STIFFZ attack the same problem, their calling sequences are identical, specifying the routine to evaluate the f_i, the y vector, N, T, the integration interval, error limit, time step, option control, error indicator, and working storage. Each has associated optional common blocks which may be used occasionally for finer control.

Finally we include, for historical reasons only, the single step fourth order Runge Kutta routine RKONE which performs one step of specified size. *This should not be used on any new programs*, as it returns no error estimate and any attempts by users to use RKONE with some error control of their own devising is extremely ill-advised.

Selection Criteria

If storage is at a premium, or the equation set is not excessively non-linear, use RKFINT. If this requires too much time, the equation set is probably stiff, so try STIFFZ with the Gear option.

If storage is available and the equation set is predominantly non-linear, use STIFFZ, Adams option. If time steps remain extremely small switch to the Gear option.

If the equation set is known to contain a large range of time constants use STIFFZ with the Gear option.

Within STIFFZ, use the full Jacobian option for $N \leq 20$ or the sparse option for $N > 20$, unless the equation set is known to be banded. For large dense equation sets, the diagonal approximation option may be the only viable choice. The nil option is a last resort when storage is restricted.

M.B. Carver
1978 June

Reference

(1) M.B. Carver, D.G. Stewart and J.L. Liu, "Evaluation, Validation and Selection of Robust Integration Algorithms for Practical Applications", Atomic Energy of Canada Limited report AECL-6220.

TITLE

Single Step Fourth Order Runge-Kutta Integration Without Error Estimate. (This routine is retained for continuity only and should not be used in new programs.)

INTRODUCTION

This routine performs one step in the integration of a set of first order differential equations of the form

$$dy_1/dy_1 = f_1 = 1$$

$$dy_n/dy_1 = f_n(y_1,y_2,y_3,\ldots,y_n) \quad 2 \le n \le N$$

(y_1 is equivalent to t in the initial value problem in the introduction to this section.)

ENTRY

CALL RKONE(H,N,AUX,VAL,DERIV,WS)

H real input variable, the interval of integration. H can be <0 if the user wishes to integrate backwards in the independent variable.

N integer input variable, the number of equations; i.e. the length of VAL, DERIV and WS.

AUX subroutine supplied by the user to calculate $f_1,f_2,\ldots,f_n$. AUX is entered from RKONE by

CALL AUX(VAL,DERIV)

Using the values of $y_1,\ldots,y_n$ in VAL(1),...,VAL(N), AUX must calculate the values of $f_2,\ldots,f_n$ and store them in DERIV(2),...,DERIV(N). It must also set DERIV(1) to 1.0.

The calling program must contain the statement

EXTERNAL AUX

VAL real array used as input to pass the starting values $y_1,y_2(y_1),\ldots,y_n(y_1)$ to RKONE. On return from RKONE, this array contains y_1+h, $y_2(y_1+h),\ldots,$ $y_n(y_1+h)$.

DERIV real array used by AUX to return values of f_n to RKONE.

WS real working storage array of length N.

ROUTINES CALLED   AUX supplied by the user.

EXIT   Control is returned to the calling routine with new y values stored in VAL.

ACCURACY   Truncation error is $0(H^5)$.

SPEED   Depends largely on the speed of AUX which is called four times by this routine.

AUTHOR   Translated from APEX by C.J. Johnson   DATE   January 1969

DOCUMENTATION REVISED BY   L.E. Evans   DATE   June 1978

TITLE Error Controlled Integration of Ordinary Differential Equations by Runge-Kutta-Fehlberg Algorithm. A reasonably small, efficient algorithm suitable for integration of most equation sets.

ENTRY CALL RKFINT (EQNSF,Y,N,T,DTINT,EPS,DTUSED,MF,INOUT,WS)

EQNSF The auxiliary subroutine to be supplied by the user and declared external in the routine which calls RKFINT. Its calling sequence is

EQNSF(N,T,Y,DY),

and it must specify the equations by defining each derivative DY(I) in equation 17-1.

Y Real input/output array, dimension N. Each element must be set to the initial value $y_i(o)$ prior to the first call to RKFINT and will thereafter contain the current value $y_i(t)$ of the dependent variables.

N Integer input variable to be set to the number of differential equations N

T Real input/output variable to be set to the initial value of the independent variable, t, prior to the first call to RKFINT, and will thereafter contain the current value of t.

DTINT Real input variable to be set to the interval in t after which RKFINT is to return to the calling program. DTINT is always taken as positive by the program. Therefore, backwards integration is not permitted.

EPS Real input variable to be set to the acceptable error tolerance required by the user and used by RKFINT to govern the step size DTUSED. A step is deemed acceptable if the estimated relative local truncation error is less than EPS for each y_i. Realistic bounds are $10^{-10} \le \text{EPS} \le 10^{-2}$.

DTUSED Real output parameter which holds the current step size used by RKFINT. The user may monitor DTUSED, but may not change it. On the first call to RKFINT, the initial trial value of DTUSED is calculated from the user's input data. On subsequent calls the last vlaue used is the one taken.

MF Integer input parameter to control error diagnostics, a non-zero value suppresses non fatal error diagnostic messages and these errors must then be assessed by examining INOUT.

INOUT Integer input/output variable, the operate flag which must be set=0 for the first call to RKFINT, and which returns a value of 1 when integration is successful. Negative values indicate that errors have occurred. (See EXIT below.)

WS Working storage array of size 8N to be reserved in the calling program.

COMMON BLOCKS USED

The common blocks need not be included in the calling routine unless the user wishes to change default values in RKFINT or more thoroughly examine the solution.

(a) COMMON/BASINT/YCUT,DYCUT,DTMAX,DTMIN

YCUT lower bound of significance of Y used for computing the relative error.

DYCUT lower bound of significance of DY used for computing the relative error.

To avoid problems approaching, leaving, or crossing zero, the relative error is

EPS*AMAX1(|Y|,YCUT,DYCUT-|DY|)

defaults are YCUT=1.0E-14,DYCUT=1.0E-9 which are appropriate for Y values of the order of unity. For very small or large Y values, default values should be modified in ratio.

DTMAX
DTMIN Upper and lower bounds permitted in DTUSED, default values 10^{15}, 10^{-15}.

(b) COMMON/STAT/KOUNT(5)

This block may be used to track progress of integration, its elements are

1) The total number of calls to EQNSF
2) The number of successful steps taken
3) The number of calls at DTMAX
4) The number of calls at DTMIN
5) The number of calls with DTUSED so small that the accuracy of the algorithm may be degraded.

Steps of type 4 and 5 are accepted, but not rated as successful.

ADDITIONAL ENTRY INFORMATION

1) The subroutine EQNSF must be provided to define the equations 17-1 as follows:

```
SUBROUTINE EQNSF(N,T,Y,DY)
Real Y(N),DY(N)
  :
DY(1) =
DY(2) =
  :
END
```

This will be called repeatedly at times selected by RKFINT to evaluate the values of DY_i in terms of Y_i and T as the integration proceeds. It may call other routines as required.

2) For the first call to RKFINT, set INOUT=0, T to T_o and Y_i to $Y_i(o)$. RKFINT will then attempt to integrate to $T_1 = T_o$ + DTINT. If it is unable to do so diagnostics will be printed if not suppressed, and INOUT will be returned negative. If the step is successful, INOUT will return +1 and the control program should arrange for printout of relevant variables, make any change required to DTINT and return to RKFINT to integrate from T_1 to T_1 + DTINT.

The actual step size DTUSED taken by the routine must not be controlled by the user other than indirectly via DTMAX, DTMIN and EPS, but it should normally be printed out to assess the progression of integration.

EXIT RKFINT returns the current values of T and Y. If INOUT returns a positive value, the Y values have been obtained within a per step relative accuracy of EPS.

Values of INOUT less than -1 indicate that the returned solutions may be inaccurate, as steps which do not satisfy the error criterion may have been accepted for the following reasons

INOUT	REASON
-1	A number of steps KOUNT(3) were taken at DTMAX so efficiency was degraded.
-2	Integration failed to satisfy the error test at DTMIN for a total number of KOUNT(4) steps.
-3	DTUSED was so small in relation to Y that the accuracy of the algorithm may have been degraded for a total number of KOUNT(5) steps.
-4	The imposed error criterion appears to be too stringent for this problem as 10 consecutive steps have failed the above two criteria.
-5	Illegal value entered for DTMIN or DTMAX, probably because of a user blunder, or core overwrite.

The latter two errors are fatal, and a subsequent entry to RKFINT with these values of INOUT causes the job to be stopped by the system as further results would be meaningless.

Appropriate messages are printed with each of the above fatal errors and unless print is suppressed by entering a negative value of MF, non fatal errors also produce messages.

STORAGE 4100_8

EXAMPLE The following routines use RKFINT to compute the solution to a set of three ordinary differential equations:

$$y_1' = 4.5(y_2-y_3) - 5.5\ y_1$$

$$y_2' = 49.5(y_1-y_3) - 50.5\ y_2$$

$$y_3' = 45.0(y_1-y_2) - 55.0\ y_3$$

with initial condition $y_1 = y_2 = y_3 = 2.0$. The calling program EG sets up parameters, performs printout and calls RKFINT. The routine EQNSF defines the equations.

```
      PROGRAM EG(OUTPUT,TAPE6=OUTPUT)
C
C        EXAMPLE PROGRAM FOR RKFINT
C
      REAL Y(3),WS(24)
      COMMON/STAT/KOUNT(5)
      EXTERNAL EQNSF
C
C        SET INITIAL VALUES AND CONTROLS
C
      PRINT 1Ø1
      T=Ø.Ø
      EPS=1.ØE-Ø5
      N=3
      MF=Ø
      INOUT=Ø
      DTINT=5.
      DO 1Ø I=1,N
 1Ø   Y(I)=2.
C
C        TRANSFER TO RKFINT
C
 2Ø   CALL RKFINT(EQNSF,Y,N,T,DTINT,EPS,H,MF,INOUT,WS)
C
C        TEST ERROR FLAG
C
      IF(INOUT.LT.Ø) GO TO 4Ø
C
C        PRINT RESULTS
C
 3Ø   WRITE(6,1ØØ) T,H,Y
C
C        CONTINUE UNTIL T = 1Ø
C
      IF(T.LT.1Ø.) GO TO 2Ø
      GO TO 5Ø
C
 4Ø   WRITE(6,11Ø) INOUT,KOUNT
C
C     ERROR FLAG LT Ø PRINT AND CONTINUE IF ERROR NON-FATAL
C
      IF(INOUT.GT.-4) GO TO 3Ø
      STOP
C
 5Ø   CONTINUE
      PRINT 12Ø
C
 1ØØ  FORMAT(* TIME*G1Ø.3* STEP*G1Ø.3* Y1*G1Ø.3* Y2*G1Ø.3* Y3*G1Ø.3)
 1Ø1  FORMAT(1H1)
 11Ø  FORMAT(* INOUT = *,I3* KOUNT =*,7I5)
 12Ø  FORMAT(* RKFINT SUCCESSFUL  FINISHED AT TIME = 1Ø.*)
      END
      SUBROUTINE EQNSF(N,T,Y,DY)
      REAL Y(N),DY(N)
C
C        DEFINE THE EQUATIONS HERE
C
      DY(1)=4.5*(Y(2)-Y(3)-Y(1))-Y(1)
      DY(2)=49.5*(Y(1)-Y(2)-Y(3))-Y(2)
      DY(3)=45.Ø*(Y(1)-Y(2))-55.*Y(3)
C
      END
```

```
TIME  5.ØØ      STEP  .29ØE-Ø1 Y1  .674E-Ø2 Y2  .674E-Ø2 Y3  .2ØØE-11
TIME  1Ø.Ø      STEP  .313E-Ø1 Y1  .454E-Ø4 Y2  .454E-Ø4 Y3 -.918E-12
RKFINT SUCCESSFUL  FINISHED AT TIME = 1Ø.
```

REFERENCES

[1] M.B. Carver, D.G. Stewart and J.L. Liu, "Evaluation, Validation and Selection of Robust Integration Algorithms for Practical Applications", Atomic Energy of Canada Limited report AECL-6220.

AUTHOR James Liu DATE June 1977

TITLE

Error Controlled Integration of Ordinary Differential Equations, using a Modified Gear Algorithm. A package of subroutines to integrate large, non linear, or stiff sets of ordinary differential equations.

ENTRY

CALL STIFFZ(EQNSF,Y,N,T,DTINT,EPS,DTUSED,MF,INOUT,WS)

EQNSF The auxiliary subroutine supplied by the user, and declared external in the routine which calls STIFFZ. Its parameter sequence is

EQNSF(N,T,Y,DY)

and it must specify each of the N differential equations by defining each of the derivatives DY(I) in equation 17-1.

Y Real input/output array of dimension N. Each element must be set to the initial value $y_i(0)$ prior to the first call to STIFFZ, and will thereafter contain current values $y_i(t)$ of the dependent variables.

N Integer input variable to be set to the number of differential equations N.

T Real input/output variable to be set to the initial value of the independent variable, t, prior to the first call to STIFFZ and will thereafter contain the current value of t.

DTINT Real input variable to be set to the interval in t after which STIFFZ is to return to the calling routine. DTINT is always taken as positive by the program. Therefore, backwards integration is not permitted.

EPS Real input variable to be set to the acceptable error tolerance required by the user, and used by STIFFZ to govern step size used, DTUSED. A step is deemed acceptable if the estimated relative local truncation error is less than EPS for each y_i. Realistic values are $1.0\ E^{-10} \leq EPS \leq 1.0E^{-2}$.

DTUSED Real output parameter which holds the current stepsize used by STIFFZ. The user may monitor DTUSED but may not change it, except in special cases where discontinuities are to be handled via the common blocks /DISCO/ and /STIFS/ as discussed below.

MF Two-digit input integer, specifying

(a) Method indicator, nm=|MF|, defined as follows:

If n=1, use Adam's Bashforth predictor corrector (for non-linear equation systems).

n=2, use Gear's stiff system predictor corrector (for stiff equation systems).

m=1, use explicit method, no Jacobian analysis.

m=2, use implicit method, full Jacobian analysis.

m=3, use implicit method, diagonal Jacobian approximation.

m=4, use implicit method, banded Jacobian analysis.

m=5, use implicit method, sparse Jacobian analysis.

The full and sparse Jacobian analyses are the most accurate and permit the greatest efficiency of the routine. Banded approximation is next, followed by the diagonal and explicit methods, unless the Jacobian happens to be truly banded, in which case m=3 is ideal. The choice is normally a trade off between efficiency and storage required in the WS array.

(b) Printing Control: Negative values of MF suppress printing of messages concerning non fatal errors.

INOUT integer input/output variable, the operate flag, which must be set = 0 for the first call of STIFFZ and which returns the current order of the method when integration is successful, or a negative value when an error has occurred. (See EXIT below.)

WS(NWS) real working storage array used by STIFFZ chiefly to hold the Jacobian matrix $\partial f/\partial y$. When STIFFZ is called with INOUT=0, WS(1) must contain the value of NWS. In all other cases WS(1) must not be modified outside STIFFZ. The dimension requirements depend on the method indicator, nm, as follows:

m=1	NWS	$\geq$ 10N+1
2		$\geq$ N(N+11)+1
3		$\geq$ 11*N+1
4		$\geq$ N(12+2ML+MU)+1
5		$\geq$ 17 + N(26+4ML)

where ML and MU are defined below.

ADDITIONAL ENTRY INFO

1) The subroutine EQNSF must be provided to define the equations(17-1) above as follows:

```
SUBROUTINE EQNSF(N,T,Y,DY)
REAL Y(N),DY(N)

DY(1)=...
DY(2)=...
    .
    .
    .
END
```

This will be called repeatedly at times T determined by STIFFZ to evaluate the derivatives DY_i in terms of Y_i and T as integration proceeds. It may call other routines as required.

2) For the first call to STIFFZ, set INOUT=0, T to T_o, WS(1) to NWS and the Y_i to $Y_i(o)$. STIFFZ will attempt to integrate from time T_0 to $T_1 = T_0$ + DTINT. If it is unable to do so, diagnostics will be printed if not suppressed and INOUT will be returned negative. If the step is successful, INOUT will be returned positive and equal to the current order of approximation used by the algorithm. The control program should then arrange for printout of relevant variables, make any change required to DTINT, and return to STIFFZ to integrate from T_1 to T_1 + DTINT.

The actual step size DTUSED taken by the routine must not be controlled by the user other than indirectly via DTMAX,DTMIN and EPS, but it should normally be printed out to assess the progression of the integration. The subroutine may integrate to a larger value of T than the printout value required by the user. If the user wishes to limit the range of T, this may be done through DTMAX, at the possible risk of some loss in efficiency, or through the use of RESTART, as explained below.

COMMON BLOCKS USED

The common blocks need not be included in the calling routine unless the user wishes to change default values in STIFFZ or more thoroughly examine the solution.

(a) COMMON/BASINT/YCUT,DYCUT,DTMAX,DTMIN,ML,MU

YCUT Lower bound of significance of Y, used for computing the relative error.

DYCUT Lower bound of significance of DY, used for computing the relative error.

To avoid problems approaching, leaving, or crossing zero the relative error is based on

EPS*AMAX1(|Y|,YCUT,DYCUT-|DY|).

Defaults are YCUT=1.0E-14,DYCUT=1.0E-9, which are appropriate for Y values of the order of unity. For very small or large Y values, default values should be modified in ratio.

DTMAX
DTMIN Upper and lower bounds permitted in DTUSED, default values 10^{15},10^{-15}.

ML,MU integer variables whose use depends on the method, nm, default values are 1.

For $m \leq 3$, ML and MU are unused.

For m=4, ML and MU are the width of the upper and lower bands excluding the diagonal.

For m=5, ML is the sparsity indicator of the Jacobian, and the total number of non-zero elements permitted is ML*N. The structure of the Jacobian is reassessed every MU evaluations in case new non-zero terms have evolved. Suggest ML=MU=10. If MU is negative, only one initial evaluation of the Jacobian structure is performed. Obviously for m=4 or 5 to be useful, the resulting Jacobian array must be considerably smaller than it would be at m=2.

(b) COMMON/STIFS/RESTART,JSTART,MAXDER

is optional but may be included to give the user finer control over known discontinuities in the definition of the equation system.

RESTART Normally when RESTART has its default value of 0, the algorithm permits T to exceed T + DTINT and then interpolates. A discontinuity normally occurs in the middle of a step size and will cause STIFFZ to adjust step size accordingly to compute the transition, but this may be done inefficiently.

If a discontinuity is to occur at a known T=T*, it is more efficient to forbid T to exceed T* by setting RESTART negative. For the call to STIFFZ immediately prior to T*, set DTUSED so that the next value is T*, and set RESTART positive. STIFFZ will then return the exact values at T*. The discontinuity may then be introduced without causing problems if the next entry is with JSTART=0. This effectively starts a new problem from initial values at T*.

JSTART JSTART is normally 1, but if the user wishes to change DTUSED, MF or EPS during a run, he must also set JSTART = -1 when the change is made. To restart the problem from current values, set JSTART = 0.

(c) COMMON/DISCO/DISC1,DISC2

This block is used for control of discontinuities which occur at arbitrary times depending on the evolution of the integration. If DISC1 is set .TRUE. by the user, STIFFZ will return once, after every successfully completed step with DISC2 set .TRUE. The user then may check the definition of discontinuity functions and direct the integration using JSTART if necessary. The theory of discontinuities is too complex to be further discussed here, but is covered adequately by references 1 and 2.

(d) COMMON/STAT/KOUNT(7)

This block may be used to track the progress in integration, its elements monitor the following:

1) The total number of calls to EQNSF
2) The number of successful steps taken
3) The number of calls to EQNSF used for Jacobian evalution
4) The number of Jacobian evaluations
5) The number of steps at DTMIN failing the error
6) The number of steps at DTMIN failing the convergence test
7) The number of steps at DTMAX

Steps at DTMIN are accepted but rated as unsuccessful.

(e) The following three common blocks are used to communicate between various modules in STIFFZ but are not required by the user

CNTROL,SPARS,INT1

ROUTINES CALLED

STIFFZ loads and calls a number of auxiliary routines, some of which may also be used independently. They are:

GEARZ	-	Gear's algorithm, all options
COSET	-	Coefficients for GEARS, all options
DECOMP	-	Decompose a full matrix, m=2
SOLVE	-	Solve equations from DECOMP, m=2
DECB	-	Decompose a banded matrix, m=4
SOLB	-	Solve equations from DECB, m=4
JAKOB	-	Determine and pack a sparse Jacobian, m=5
SPARSE	-	Decompose a sparse matrix, m=5
SPARSEB	-	Solve equations from SPARSE, m=5
SORTAG	-	Sort an array of numbers, m=5
MSCALE	-	Scaling routine for SPARSE, m=5

In the event that a particular option is decided upon, one may prevent the unwanted routines from loading by including dummy subroutines of the same name in the user's deck.

STORAGE

$12,000_8$ including all routines.

EXIT

STIFFZ returns the current values of T and Y. If INOUT returns a positive value, this is the current order used by the method (maximum 5), and the Y values have been obtained within a per step relative accuracy of EPS.

INOUT less than -1 indicates that the returned solutions may be inaccurate, as steps which do not satisfy error criteria may have been accepted for the following reasons:

INOUT	REASON
-1	A number of steps (KOUNT(7)) were taken at DTMAX so efficiency has been degraded.
-2	Integration failed to satisfy the error test at DTMIN at a total number of KOUNT(5) steps.
-3	Corrector convergence was not achieved at DTMIN at a total number of KOUNT(6) steps.

-4 Trouble with the sparse matrix option, but sparsity was detected to be decreasing, further matrix restructuring was prohibited. The user should increase ML and NWS.

-5 The error criterion imposed appears to be entirely too strict for this problem as 10 consecutive steps have generated INOUT = -2 or -3 having failed the above criteria

-6 Illegal value of N, EPS, or MF probably due to a user blunder or over-write.

-7 The declared working storage contains insufficient room for the requested Jacobian option. If this occurs at T=0, user should change options or increase NWS. It may also occur in the sparse matrix option when sparsity decreases are not detected in advance. In this case increase ML and NWS. This may also indicate a problem with the definition of WS(1). See previous notes.

The last three problems are fatal, and if STIFFZ is entered again the program is stopped by the system as further results would be meaningless. Appropriate non-fatal error messages are printed unless MF is entered negative to suppress printing. Messages are always printed for fatal errors.

EXAMPLE The following routines use STIFFZ to compute the solution to a set of three ordinary differential equations:

$$y'_1 = 4.5(y_2-y_3) - 5.5\ y_1$$

$$y'_2 = 49.5(y_1-y_3) - 50.5\ y_2$$

$$y'_3 = 45.0(y_1-y_2) - 55.0\ y_3$$

with initial condition $y_1 = y_2 = y_3 = 2.0$. The calling program EG sets up parameters, performs printout and calls STIFFZ. The routine EQNSF defines the equations.

```
      PROGRAM EG(OUTPUT,TAPE6=OUTPUT)
C
C        EXAMPLE PROGRAM FOR STIFFZ
C
      DIMENSION Y(13),WS(155)
      COMMON/STAT/KOUNT(7)
      COMMON/BASINT/YCUT,DYCUT,DTMAX,DTMIN,ML,MU
      EXTERNAL EQNSF
C
C        SET INITIAL VALUES AND CONTROLS
C
      PRINT 1Ø1
      T=Ø.Ø
      EPS=1.ØE-Ø5
      N=3
      MF=25
      INOUT=Ø
      DTINT=5.
      ML=5
      MU=5
      WS(1)=155.
      DO 1Ø I=1,N
 1Ø   Y(I)=2.
C
C        TRANSFER TO STIFFZ
C
 2Ø   CALL STIFFZ(EQNSF,Y,N,T,DTINT,EPS,H,MF,INOUT,WS)
C
C        TEST ERROR FLAG IF LT Ø STOP
C
      IF(INOUT.LT.Ø) GO TO 4Ø
C
C        PRINT RESULTS
C
 3Ø   WRITE(6,1ØØ) T,H,(Y(I),I=1,N)
C
C        CONTINUE UNTIL T = 1Ø
C
      IF(T.LT.1Ø.) GO TO 2Ø
      GO TO 5Ø
C
C     ERROR FLAG LT Ø PRINT INOUT AND KOUNT THEN STOP
 4Ø   WRITE(6,11Ø) INOUT,KOUNT
      STOP
C
C
 5Ø   CONTINUE
      PRINT 12Ø
C
 1ØØ  FORMAT(* TIME*G1Ø.3* STEP*G1Ø.3* Y1*G1Ø.3* Y2*G1Ø.3* Y3*G1Ø.3)
 1Ø1  FORMAT(1H1)
 11Ø  FORMAT(* INOUT = *,I3* KOUNT =*,7I5)
 12Ø  FORMAT(* STIFFZ SUCCESSFUL  FINISHED AT TIME = 1Ø.*)
      END
      SUBROUTINE EQNSF(N,T,Y,DY)
      REAL Y(N),DY(N)
C
C        DEFINE THE EQUATIONS HERE
C
      DY(1)=4.5*(Y(2)-Y(3)-Y(1))-Y(1)
      DY(2)=49.5*(Y(1)-Y(2)-Y(3))-Y(2)
      DY(3)=45.0*(Y(1)-Y(2))-55.*Y(3)
C
      END
```

```
TIME  5.00     STEP  .111      Y1  .674E-02 Y2  .674E-02 Y3  .252E-14
TIME  10.0     STEP  .148      Y1  .454E-04 Y2  .454E-04 Y3 -.617E-16
STIFFZ SUCCESSFUL  FINISHED AT TIME = 10.
```

REFERENCES

[1] M.B. Carver, "Efficient Handling of Discontinuities in Ordinary Differential Equation Simulation" in press for Mathematics & Computers in Simulation, 1978.

[2] M.B. Carver and S.R. MacEwan, "Simulation of an Implicitly defined Differential Equation System Subject to Numerous Discontinuities", in press for Applied Mathematical Modelling, 1978.

AUTHORS

STIFFZ was written by M.B. Carver and D.G. Stewart and incorporates modifications of the GEAR routines of A.C. Hindmarsh, LRL, and the sparse matrix routines of A.C. Curtis and J.K. Reid, Harwell.

DATE

July 1978

1-18 INTEGRATION OF PARTIAL DIFF EQNS (FOR FUTURE USE)

2- 1 PLOTTING

00-19 BASIC PLOTTING ROUTINES

2-1-00	PLOT	PLOTM PLOTC BASIC PLOTTING ROUTINE
2-1-01	PLOGX	PLOGXM BASIC SEMI-LOG X PLOTTING ROUTINE
2-1-01	PLOGY	PLOGYM BASIC SEMI-LOG Y PLOTTING ROUTINE
2-1-01	PLOGXY	PLOGXYM BASIC LOG-LOG PLOTTING ROUTINE
2-1-02	VARSIZM	VARSIZ PLOT TEXT WITH TWO CHARACTER SIZES
2-1-03	PLODAM	PLODA PLOT DASHED LINES, OPTIONAL SMOOTHING
2-1-10	AXLIN	AXLINM LINEAR AXES LABELLING ROUTINE
2-1-10	AXLGX	AXLGXM SEMI-LOG X AXIS LABELLING ROUTINE
2-1-10	AXLGY	AXLGYM SEMI-LOG Y AXIS LABELLING ROUTINE
2-1-10	AXLGXY	AXLGXYM LOG-LOG X AND Y AXES LABELLING ROUTINE
2-1-11	SMLOGX	SMLOGXM PLOT AND LABEL AXES FOR SEMI-LOG X PLOTS
2-1-11	SMLOGY	SMLOGYM PLOT AND LABEL AXES FOR SEMI-LOG Y PLOTS
2-1-12	LOGLOG	LOGLOGM PLOT AND LABEL AXES FOR LOG-LOG PLOTS

20-29 ROUTINES PRODUCING FIGURES OR COMPLETE PLOTS

2-1-20	SIMPLT	SIMPLTM DUMPON DUMPOFF SIMPLE PLOTTING ROUTINE

30-39 THREE-DIMENSIONAL AND CONTOUR PLOTTING ROUTINES

2-1-30	SPACE	SPACEM THREE-DIM PERSPECTIVE SURFACE PLOTTING
2-1-31	PLOT3D	PLOT3DM THREE-DIM SURFACE PLOTTING
2-1-35	CONTOUR	CONTOUM CONTOUR SURFACE PLOTTING

40-49 PRINTER PLOT ROUTINES

2-1-40	PROT	PRINTER PLOTTING ROUTINE

50-59 PLOTTER CONTROL

2-1-50	SPLOT	POSITION PLOTTER PEN ON GRAPH PAPER

60-69 CALCOMP PLOTTING ROUTINES

2-1-60	PLOTCI	PLOTI CALCOMP INITIALIZATION ROUTINE
2-1-61	FACTOR	SET CALCOMP PLOTTING FACTOR
2-1-62	WHERE	RETRIEVE CALCOMP FACTOR AND COORDINATES
2-1-63	PLT	BASIC CALCOMP PLOTTING ROUTINE
2-1-64	SYMBOL	PLOT TEXT AND SPECIAL SYMBOLS
2-1-65	NUMBER	PLOT FLOATING POINT NUMBERS
2-1-66	SCALE	CALCOMP SCALING ROUTINE
2-1-67	AXIS	CALCOMP AXIS PLOTTING ROUTINE
2-1-68	LINE	CALCOMP LINE PLOTTING ROUTINE

2-1 PLOTTING

Introduction

The plotting facilities at CRNL were developed locally and are most likely quite different from those at other installations. Plotting is accomplished in two distinct steps:

1. A user's plotting program creates an output file of plotting data.

2. On job termination, the 3300 system uses this file to produce plotted output on a Calcomp plotter. (Suitably modified plot files are also used to produce plotted output on graphical display terminals.)

This file of data, called PLOT, is a file associated with the user's job and must therefore be declared on the program card along with INPUT, OUTPUT and any other files being used. The low level plotting routine, PLOT, is the only routine that actually generates the data for Step 1. All other routines, except for the printer plot routine PROT, eventually call PLOT.

The basic features of our plotting system (i.e. of PLOT) are:

i) definition of a drawing area

ii) drawing axes

iii) drawing a line or a set of points with the option of using special symbols

iv) drawing a string of text.

The write-up of PLOT describes these features in detail.

Most AELIB routines allow either English or metric units to be used to specify physical plotting dimensions or coordinates. To do this, these routines have two entry points, one assuming English units (inches) and the other assuming metric units (millimetres). In most cases plotting coordinates are not specified in either system but as numbers defined by the user's program (or what we call user units). The subroutine PLOT automatically scales such user unit coordinates to the drawing area desired. (The drawing area is defined by calling PLOT or PLOTM with MODE=1 or MODE=10.) Even if user units are used, all plotting on a given drawing area must be done by either all metric or all English plotting calls. If English and metric calls are mixed, an error message is printed and the job is terminated.

The six groups of plotting routines will now be briefly discussed:

2-1 PLOTTING

Basic Plotting Routines

In addition to the basic routine PLOT, there are three routines for logarithmic plotting; PLOGX for semi-log plotting with X axis logarithmic, PLOGY for semi-log plotting with Y axis logarithmic and PLOGXY for log-log plotting. The general purpose axis labelling routines AXLIN, AXLGX, AXLGY and AXLGXY are to be used with PLOT, PLOGX,PLOGY and PLOGXY (respectively). We recommend these routines for low-level plotting.

The extra logarithmic plotting routines SMLOGX,SMLOGY and LOGLOG plot and label the axes in one call but the data must be plotted using PLOT. This means that the user's program must calculate the necessary logarithms.

VARSIZM is an extension of the basic plotting routine, PLOT, that plots character strings using two character sizes. Note that VARSIZM requires Hollerith string input while PLOT with MODE=3 requires a variable format with optional variable list.

PLODAM is a routine for plotting dashed lines but may also be used to plot smooth curves through data points. It uses the IMSL routine ICSICU to fit a smooth curve to the data and then plots this curve using a dashed line specified by the user.

Routines Producing Complete Figures or Plots

SIMPLT is a general purpose subroutine which will produce complete linear, semi-log or log-log plots. Although SIMPLT is used mainly to get a quick look at a set of data points, this subroutine allows the user extensive control over the appearance of the resulting plot.

The write-up has been prepared with the basic calling information first. Only those users wishing to modify the default SIMPLT format should read beyond the first program example.

Three Dimensional and Contour Plotting Routines

All three routines in this section, SPACE,PLOT3D and CONTOUR, process a single valued function (i.e. a surface) defined on a rectangular x-y grid. SPACE plots a perspective view of this surface with or without a three dimensional frame while PLOT3D plots a parallel projection of the surface only. Some flexibility of surface orientation is provided with each routine.

CONTOUR plots a contour map of the surface for a given set of equally spaced contour values. The user must define the plotting surface (using PLOT) before calling CONTOUR. The necessary frame is set up automatically by SPACE and PLOT3D.

There is a possibility of some confusion about the axes and definition of the surface used in these routines. SPACE and PLOT3D assume a three dimensional axis system as in Fig. 1, whereas CONTOUR uses one as in Fig. 2.

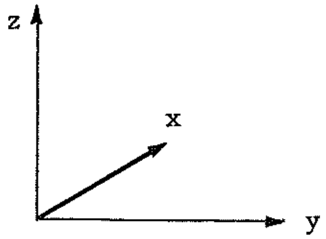


Fig. 1

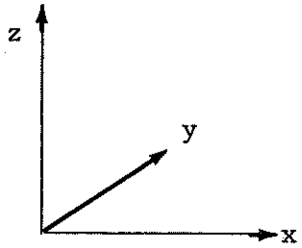


Fig. 2

To use CONTOUR interchangeably with SPACE or PLOT3D, the roles of the formal parameters x and y must be reversed.

Printer Plotting

The printer plot routine PROT plots up to three curves per frame and is adequate for our very limited printer plotting requirements.

Special Plotter Control

If special ruled forms, for example, graph paper, are to be used for plotting, SPLOT should be used to allow the operators to set the initial pen position.

Calcomp Plotting Subroutines

The subroutines in this group provide very nearly the same plotting capability as the standard Calcomp subroutines of the same names which are supported at other computer installations. This means that programs coming from or being sent to such installations can produce plotted output with very little conversion effort. Also, these routines support some features not easily provided by our other plotting software based on PLOT. Two of these are:

i) independent control over the plotting of axes via the routine AXIS; thus non-standard and multiple axes are easily produced.
ii) automatic generation of formats for real or integer numbers using the routine NUMBER.

Note, however, that programs should be written to use either Calcomp routines or our other plotting software. Mixing calls to these two systems is usually fatal.

The coordinate system used by the Calcomp routines is quite different from PLOT's. A brief description follows:

The coordinate system used in Calcomp plotting routines is rectangular, is based on units of inches, and is characterized by a variable plotting origin (x_0,y_0) and a variable plotting size factor, f. The pen coordinates (x,y) specified by the user are scaled by the plotting factor and translated from the origin to yield an actual pen position, in inches, of $(x*f+x_0, y*f+y_0)$.

Both the origin and factor are under program control. The origin may be changed by a special call of PLT and the factor may be changed by calling FACTOR. The default values for the origin and factor are (0,0) and 1 respectively. Below are three sets of pen coordinates and their equivalent pen positions with different origins and factors in effect.

User Specified Pen Coordinates	Actual Pen Position (in inches) on Plotting Surface		
	Origin (0,0) Factor 1	Origin (2,1) Factor 1	Origin (2,1) Factor .5
(0,0)	(0,0)	(2,1)	(2,1)
(4,3)	(4,3)	(6,4)	(4,2.5)
(-1,1)	(-1,1)	(1,2)	(1.5,1.5)

2-1 PLOTTING

Points To Note

i) Standard Calcomp plotting requires that the current pen position be known to and maintained by the Calcomp plotting routines. This requirement is supported locally by using as the current pen position a value determined from the previous position and the last operation performed. Position assumptions used are contained in the writeup of the routine WHERE. Each time a Calcomp routine requests that the pen be moved, the pen position is updated.

ii) The routines PLT, NUMBER, SYMBOL and AXIS all allow the programmer to specify either or both pen coordinates as the coordinates of the current pen position by using the special pen coordinate 999. This option is useful when plotting character strings with variable constituents or when labelling plotted curves.

Because the Calcomp routines are implemented by calls to our local routine PLOT, there are some local restrictions to be noted:

i) Character sizes and orientation of character strings are limited to those supported by our local PLOT. (See NUMBER,SYMBOL or AXIS writeup for details.)

ii) Special symbol set is limited to that supported by PLOT.

iii) The plotting surface must be initialized by calling PLOTCI. The Calcomp standard buffer initialization routine PLOTS is not required or supported here.

iv) Our version of the Calcomp routine PLOT is called PLT. (Programs need not be modified for this since the loader can do the substitution.)

L.E. Evans
April 1978
Revised by G.N. Williams
July 1979

TITLE Low Level Plotting Subroutine

INTRODUCTION This subroutine provides all the basic plotting features of which the following are the most often used:

a) define a drawing area
b) set up axes
c) draw a line or series of points anywhere in the drawing area, and
d) output a string of text in the drawing area.

This subroutine has three entry points: PLOT (for English units), PLOTM (for metric units) and PLOTC (used by the Calcomp routines). PLOTC calls are identical to those for PLOT.

Note that this subroutine produces a file called PLOT that is subsequently processed to produce plotted output. The handling of this file is incompatible with FORTRAN 5 OPEN and CLOSE instructions.

ENTRIES Note that several of the arguments of PLOT may be of type INTEGER or REAL. These are marked with an asterisk(*); all other arguments must be of type REAL.

1) <u>Define a Drawing Area, Calculate Scale Factors and Set Up Axes</u>

CALL $\left\{\begin{matrix}\text{PLOT}\\\text{PLOTM}\end{matrix}\right\}$ $(\overset{*}{\text{MODE}},\overset{*}{\text{XL}},\overset{*}{\text{YL}},\text{XMIN},\text{XMAX},\text{YMIN},\text{YMAX},\overset{*}{\text{TICX}},\overset{*}{\text{TICY}})$

MODE =1

XL,YL lengths of the axes in inches (PLOT) or millimetres (PLOTM).

for PLOT $.005 \le XL \le 160.0$
$.005 \le YL \le 28.175$

for PLOTM $.127 \le XL \le 4064.$
$.127 \le YL \le 715.645$

If YL > 10.175 (PLOT) or 258.445 (PLOTM), the wide plotter is automatically selected. Otherwise, the narrow plotter is used.

$\left.\begin{matrix}\text{XMIN,XMAX}\\\text{YMIN,YMAX}\end{matrix}\right\}$ real, bounds on variable values for x and y axes. XMIN(YMIN) may be larger than XMAX(YMAX) if desired.

TICX =-1, scale factors are calculated, the drawing area is defined, but no axes are drawn.

=0, lines of length XL are drawn at YMIN,YMAX.

>0, lines of length XL are drawn at YMIN,YMAX and ticks are drawn on these lines at those points X = ±N x TICX, N=0,1,2,... which fall within the interval XMIN $\leq$ X $\leq$ XMAX.

TICY Analogous to TICX.

Note: A MODE 1 call of PLOT(PLOTM) is converted into

CALL $\begin{Bmatrix}\text{PLOT}\\ \text{PLOTM}\end{Bmatrix}$ (10,XD,YD)

CALL $\begin{Bmatrix}\text{PLOT}\\ \text{PLOTM}\end{Bmatrix}$ (11,XO,YO,XL,YL,XMIN,XMAX,YMIN,YMAX,TICX,TICY)

where

for PLOT:
XD=XL+3.0
YD=28.675 if YL>10.175
=10.675 if YL$\leq$10.175
XO=2.0
YO=0.5

for PLOTM:
XD=YL+76.2
YD=728.345 if YL>258.445
=271.145 if YL$\leq$258.445
XO=50.8
YO=12.7

2) Plot a Set of Points

CALL $\begin{Bmatrix}\text{PLOT}\\ \text{PLOTM}\end{Bmatrix}$ ($\overset{*}{\text{MODE}}$,$\overset{*}{\text{PEN}}$,$\overset{*}{\text{SYMB}}$,XX,YY,$\overset{*}{\text{N}}$)

MODE =2

PEN =0, pen is raised between points.
=1, pen is left on paper between points.

SYMB symbol to be plotted at each point $0 \leq \text{SYMB} \leq 24$. Values outside this range revert to no symbol. Symbols generated by odd numbers in the range 1 to 19 are 1 mm (.04") in size; symbols generated by even numbers in the range 2 to 20 and by numbers 21 and 23 are 2 mm (.08") in size; symbols generated by numbers 22 and 23 are 4 mm (.16") in size.

SYMB	Symbol	SYMB	Symbol
0	No symbol	13,14	▷•
1, 2	□	15,16	▽
3, 4	△	17,18	◁
5, 6	+	19,20	◇
7, 8	✕	21,22	⧗
9,10	⫯	23,24	⋈
11,12	-•-		

XX,YY real arrays containing the (x,y) coordinates of the points to be plotted, in the user's units.

N integer variable, where |N| is the number of points to be plotted.

If N>0, coordinate pairs are plotted in the order (XX(1),YY(1));...;(XX(N),YY(N))

If N<0, coordinate pairs are plotted in the order (XX(N),YY(N));...;(XX(1),YY(1))

3) Plot a String of Text

CALL $\left\{\begin{array}{l}\text{PLOT}\\ \text{PLOTM}\end{array}\right\}$ $(\overset{*}{\text{MODE}},\overset{*}{\text{FORM}},\overset{*}{\text{SIZE}},\overset{*}{\text{DIR}},\text{XLOC},\text{YLOC},\overset{*}{\text{FW}},\overset{*}{\text{N}})$

or

CALL $\left\{\begin{array}{l}\text{PLOT}\\ \text{PLOTM}\end{array}\right\}$ $(\overset{*}{\text{MODE}},\overset{*}{\text{FORM}},\overset{*}{\text{SIZE}},\overset{*}{\text{DIR}},\text{XLOC},\text{YLOC})$

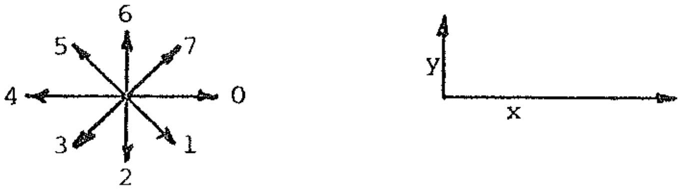
MODE =3 or 13

FORM variable or array used to transmit a format. PLOT will process FORM only up to the first line terminator. Additional specifications are ignored.

SIZE size of characters to be plotted, $1 \le \text{SIZE} \le 10$. Values outside this range will be replaced by 3. The size of the characters is SIZE x 2 mm (.08") if DIR is even and SIZE x √2 x 2 mm (.08") if DIR is odd.

If SIZE is real valued, the next lowest integer value will be used.

DIR specifies the direction of the character line.



XLOC, YLOC real variables, specify the position where the line of characters is to begin (lower left corner of the square enclosing the first character).

For MODE=3, XLOC,YLOC must be in user units. For MODE=13, XLOC,YLOC must be in inches (PLOT) or millimeters (PLOTM), measured from XO,YO, the origin specified in the last mode 11 call.

FW real or integer variable or array containing the variable value(s) to be plotted under the format passed in FORM. The type of elements in FW must match the format descriptions in FORM.

N number of data items to be used from FW.

4) Return Parameters Passed in Last MODE1 or MODE11 Call of PLOT

```
                 *
CALL (PLOT  ) (MODE,XLR,YLR,XMINR,XMAXR,YMINR,YMAXR,
     (PLOTM )  TICXR,TICYR)
```

MODE = 4

The other arguments store the previous MODE 1 or MODE11 arguments.

5) Send Messages to the 3300 Console

```
                 *     *    *
CALL (PLOT  ) (MODE,FORM,FW,N)
     (PLOTM )
```

or

```
                 *     *
CALL (PLOT  ) (MODE,FORM)
     (PLOTM )
```

MODE = 5

FORM variable or array to transmit a format.

FW real variable or first word of a real array of variables to be typed at the 3300 console under the format passed in FORM.

N number of variables to be used from FW.

6) Specify Time Limit or Disposition of File PLOT

```
                 *    *   *     *
CALL (PLOT  ) (MODE,I,ITIME,NOW)
     (PLOTM )
```

MODE = 6

I specifies disposition information for file PLOT as follows:

=0, no disposition information given.

=1, special plotter instructions (supplied by user on a job slip).

=6, request wide plotter (also done automatically if YL > 10.75 (PLOT) or 258.445 (PLOTM) in MODE1 call, or if YD > 10.675 (PLOT) or 271.145 (PLOTM) in MODE10 call).

Any other value, same as I=0.

ITIME — This parameter is ignored at present. It is retained for compatibility with previous versions of PLOT.

NOW — specifies time of release of PLOT file as follows:

=0, plot file to be released at end of job (default)
=1, whatever is in plot file to be released now.

Notes:

(a) If I=0 and NOW=0, no action will be taken.
(b) If NOW=1, the next call to PLOT must have MODE=1.

7) <u>Close or Evict the File PLOT</u>

$$\text{CALL}\begin{Bmatrix}\text{PLOT}\\ \text{PLOTM}\end{Bmatrix}(\overset{*}{\text{MODE}},\text{I})$$

MODE = 7

I — =0, close plot file and flush buffers. This is normally done automatically at the end of FORTRAN execution and need not be done by the user.

=1, evict plot file. This destroys the plot file and no plotting occurs. The user may then create a new plot file.

8) <u>Define a Drawing Area</u>

$$\text{CALL}\begin{Bmatrix}\text{PLOT}\\ \text{PLOTM}\end{Bmatrix}(\overset{*}{\text{MODE}},\overset{*}{\text{XD}},\overset{*}{\text{YD}})$$

MODE = 10

XD,YD dimensions of the drawing area in inches (PLOT) or millimetres (PLOTM).

for PLOT $.005 \le \text{XY} \le 163.0$
$.005 \le \text{YD} \le 28.675$

for PLOTM $.127 \le \text{XD} \le 4140.2$
$.127 \le \text{YD} \le 728.345$

If YD > 10.675 inches (PLOT) or 271.145 mm (PLOTM), the wide plotter is selected automatically. Otherwise, the narrow plotter is used.

Notes:

(1) A MODE10 call of PLOT(PLOTM) always sets up a new frame and must precede any other call to PLOT(PLOTM) (except a MODE1 call).

(2) A MODE10 call can be regarded as containing within it an imaginary MODE11 call of the form:

$$\text{CALL} \left\{\begin{matrix}\text{PLOT}\\ \text{PLOTM}\end{matrix}\right\}(11,0.0,0.0,XD,YD,0.0,XD,0.0,YD,-1,-1)$$

which initializes the user's units to inches (PLOT) or millimetres (PLOTM).

9) Calculate Scale Factors and Set Up Axes

$$\text{CALL} \left\{\begin{matrix}\text{PLOT}\\ \text{PLOTM}\end{matrix}\right\}(\overset{*}{\text{MODE}},\text{XO},\text{YO},\overset{*}{\text{XL}},\overset{*}{\text{YL}},\text{XMIN},\text{XMAX},\text{YMIN},\text{YMAX},\overset{*}{\text{TICX}},\overset{*}{\text{TICY}})$$

MODE = 11

XO,YO real, coordinates of the axes frame origin in inches (PLOT) or millimetres (PLOTM) measured from lower left corner of drawing area.

XL,YL lengths of the axes, in inches (PLOT) or millimetres (PLOTM). The axes must fit completely within the drawing area defined in the last mode 10 call; i.e.

$\text{XO} + \text{YL} \le \text{XD}$, $\text{YO} + \text{YL} \le \text{YD}$

XMIN, XMAX — real, bounds on variable values for x axis.

YMIN, YMAX — real, bounds on variable values for y axis.

TICX — tick flag for x axis.

=-1, scale factors are calculated but no axis is drawn.

=0, lines of length XL are drawn at YO and YO+YL.

>0, lines of length XL are drawn at YO and YO+YL, and ticks are drawn on these lines at those points X = ±N*TICX, N=0,1,2 which fall within the interval XMIN ≤ X ≤ XMAX.

TICY — tick flag for y axis, values are analogous to TICX.

Note: A MODE11 call resets the user's units so that x-coordinates in the range (XMIN,XMAX) are mapped onto the drawing area region (XO,XO+XL) and similarly for the y-coordinates.

NOTES

A MODE10 call defines a drawing area, and a MODE11 call defines a frame within that area. Therefore,

(a) The first call to PLOT must be either a MODE1 or MODE10 call.

(b) All subsequent calls to PLOT will refer to this drawing area until a new drawing area is defined in another MODE10 (or 1) call.

(c) MODE2 and MODE3 calls are drawn using the user's units defined in the last MODE11 (1) call.

(d) MODE13 calls are drawn with respect to the axes frame orgin defined in the last MODE11(1) call.

(e) Any number of MODE11 calls may appear within one drawing area. The user is responsible for any overlap of axes frames.

EXIT

Normal Exit: If no errors are detected by PLOT, then control is returned to the calling routine.

Error Exits: Any error detected by PLOT will cause the printing of a diagnostic message and the termination of the job with a FTN error number 52. The following is a list of error messages with error condition explanation where necessary:

```
***BAD MODE PARAMETER IN CALL TO PLOT or PLOTM
     The mode parameter has a value other than those
     allowed.
***FIRST CALL TO PLOT or PLOTM MUST BE MODE1
***ZERO, NEGATIVE OR HUGE VALUE FOR XL
***ZERO, NEGATIVE OR HUGE VALUE FOR YL
     XL(XD,XO) and YL(YD,YO) on a MODE1(MODE11) call must
     be within the range specified for MODE1(MODE11)
     call.
***X OR Y MAX AND MIN VALUES EQUAL
***FIRST 40 CHARACTERS OF MESSAGE ARE BLANK
     This error is detected in a MODE5 call.
***ILLEGAL NUMBER OF PARAMETERS
***ERROR IN MODE6 CALL - BAD DISPOSITION PARAMETER
     The disposition parameter has a value other than
     those allowed.
***ATTEMPT TO RELEASE EMPTY PLOT FILE
     This error is detected in a MODE6 call.
***ILLEGAL VALUE FOR ... IN CALL TO PLOT or PLOTM
     All parameters are checked by PLOT or PLOTM.  This
     error message reflects the first indefinite or
     infinite value found.
***ERROR IN MODE1(MODE11) CALL - BADLY SCALED PLOT
***ZERO VALUE FOR NUMBER OF X,Y PAIRS
***CAN ONLY CHANGE SYSTEM OF DIMENSIONS ON MODE1
CALL OF PLOT OR PLOTM
     This error results from mixing English and metric
     plot calls on the same frame.
***ERROR IN ATTEMPT TO SET TIME LIMIT.  SEE DAYFILE.
```

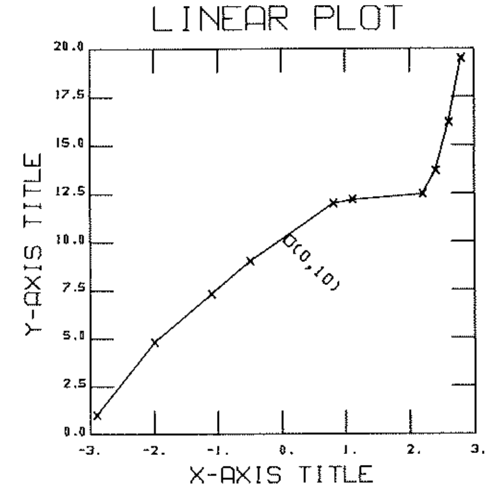
COMMON BLOCKS USED   AELERCM.

STORAGE REQUIRED 3000_8

EXAMPLE   The following program illustrates the use of PLOT. The AELIB routine AXLIN is used to plot centred axis titles and tick labels on the graph. (The plotted output has been reduced to half size for publication.)

```
      PROGRAM LINPLT(INPUT=/8Ø,OUTPUT,PLOT)
      REAL X(1Ø),Y(1Ø),FORM(2)
      INTEGER IX(2)
      DATA N/1Ø/
      DATA X/-2.9,-2.Ø,-1.1,-.5,.8,1.1,2.2,2.4,2.6,2.8/
      DATA Y/1.Ø,4.8,7.3,9.Ø,12.Ø,12.2,12.5,13.7,16.2,19.5/
      XL=4.Ø
      YL=4.Ø
      XMIN=-3.Ø
      XMAX=3.Ø
      YMIN=Ø.Ø
      YMAX=2Ø.Ø
      TICX=1.Ø
      TICY=2.5
C
C     PLOT A GRAPH WITH X-AXIS RUNNING FROM -3 TO 3 AND Y-AXIS FROM Ø TO
C     2Ø. POSITION TICKS AT INTEGER VALUES ALONG X-AXIS, AND AT MULTIPLES
C     OF 2.5 ALONG Y-AXIS.
C
      CALL PLOT(1,XL,YL,XMIN,XMAX,YMIN,YMAX,TICX,TICY)
C
C     PLOT CENTERED AXIS TITLES AND TICK LABELS ON THE GRAPH.
C
      CALL AXLIN(16H("X-AXIS TITLE"),12,16H("Y-AXIS TITLE"),12,6H(F3.Ø),
     ,3,6H(F4.1),4)
      FORM(1)=1ØH("LINEAR P
      FORM(2)=1ØHLOT")
      SIZE=3.Ø
      DIR=Ø.Ø
      XLOC=Ø.68
      YLOC=4.2
C
C     PLOT TITLE "LINEAR PLOT" ABOVE THE GRAPH.
C
      CALL PLOT(13,FORM,SIZE,DIR,XLOC,YLOC)
      PEN=1.Ø
      SYMB=8.Ø
C
C     PLOT DATA POINTS
C
```

```
      CALL PLOT(2,PEN,SYMB,X,Y,N)
      FORM(1)=1ØH("O(",I1,1
      FORM(2)=1ØHH,,I2,1H))
      SIZE=1.Ø
      DIR=1.Ø
      XLOC=IX(1)=Ø.
      YLOC=IX(2)=1Ø.
C
C     PLOT AND LABEL THE POINT (Ø,1Ø)
C
      CALL PLOT(3,FORM,SIZE,DIR,XLOC,YLOC,IX,2)
      STOP
      END
```



AUTHOR C.J. Tanner DATE December 1975
C.J. Tanner July 1979

TITLE Logarithmic Plotting Subroutines:

PLOGX or PLOGXM for x axis logarithmic
PLOGY or PLOGYM for y axis logarithmic
PLOGXY or PLOGXYM for both axes logarithmic

ENTRIES Each of these subroutines has two entry points, one for English units and one for metric units. PLOGX,PLOGY and PLOGXY assume English units and PLOGXM, PLOGYM and PLOGXYM assume metric units. As with PLOT, arguments marked with an asterisk (*) may be INTEGER or REAL.

1) Calculate Scale Factors and Plot the Frame for the Graph

```
              *                                   *    *
      ( PLOGX(MODE,XL,YL,XMIN,XMAX,YMIN,YMAX,LGTICX,TICY)
      | PLOGXM
      |       *                               *       *
CALL  < PLOGY(MODE,XL,YL,XMIN,XMAX,YMIN,YMAX,TICX,LGTICY)
      | PLOGYM
      |        *                                  *        *
      | PLOGXY(MODE,XL,YL,XMIN,XMAX,YMIN,YMAX,LGTICX,LGTICY)
      ( PLOGXYM
```

MODE =1 (can be either real or integer).

XL,YL Real, lengths of the axes

For PLOGX,PLOGY and PLOGXY:

$.005 \le XL \le 160.0$
$.005 \le YL \le 28.175$

For PLOGXM,PLOGYM and PLOGXYM:

$.127 \le XL \le 4064.$
$.127 \le YL \le 715.645$

XMIN,XMAX YMIN,YMAX Real, bounds on variables. (XMIN,YMIN) may be larger than (XMAX,YMAX) if desired. If this is the case, the plot will be scaled in descending order from left to right or bottom to top for the x-axis or y-axis, respectively.

TICX Either real or integer, used if x-axis is linear.

<0 scale factors are calculated, the plotting area is defined, but no frame is drawn.

=0 as above, but frame is drawn; i.e. lines of length XL are drawn at YMIN and YMAX or $\log_{10}$ (YMIN) and $\log_{10}$(YMAX), whichever is appropriate.

>0 as with "=0" above, plus ticks are drawn along frame at those points x = ±N*TICX (N=0,1,2,3,...) which fall on or between XMIN and XMAX.

LGTICX Either real or integer, used if x-axis is logarithmic.

=-1 scale factors are calculated, the plotting area is defined, but no frame is drawn.

=0 as above, but frame is drawn; i.e. lines of length XL are drawn at YMIN and YMAX or $\log_{10}$ (YMIN) and $\log_{10}$ (YMAX), whichever is appropriate.

>0 as with "=0" above, plus lines or ticks are drawn at those values of x as determined from the following table:

LGTICX	Result
1	lines drawn at x = 1 x 10^N (for all appropriate N).
2	lines drawn at x = 1 x 10^N, and 5 x 10^N.
3	lines drawn at x = 1 x 10^N, 2 x 10^N, and 5 x 10^N.
4	lines drawn at x = {1,2,4,6 and 8}*10^N.
5	lines drawn at x = {1,2,3,4,5,6,7,8 and 9} x 10^N.
11	ticks drawn at x = 1 x 10^N (for all appropriate N).
12	ticks drawn at x = 1 x 10^N, and 5 x 10^N.
13	ticks drawn at x = 1 x 10^N, 2 x 10^N, and 5 x 10^N.

14 ticks drawn at x = {1,2,4,6, and 8} x 10^N.

15 ticks drawn at x = 1{1,2,3,4,5,6,7,8, and 9} x 10^N.

Any other value LGTICX = 3 assumed

TICY
LGTICY Analogous to TICX,LGTICX, but for the y-axis.

Note: A MODE1 call of PLOGzz always sets up a new frame and must precede all corresponding MODE 2, 3, or 4 calls of PLOGzz.

2) Plot a Set of Points

$$\text{CALL}\left\{\begin{matrix}\text{PLOGX}\\ \text{PLOGXM}\\ \text{PLOGY}\\ \text{PLOGYM}\\ \text{PLOGXY}\\ \text{PLOGXYM}\end{matrix}\right\}(\overset{*}{\text{MODE}},\overset{*}{\text{PEN}},\overset{*}{\text{SYMB}},\text{XX},\text{YY},\overset{*}{\text{N}})$$

MODE =2 (can be either real or integer).

PEN,SYMB Either real or integer; same as in MODE2 call of PLOT or PLOTM.

XX,YY Real arrays containing the (X,Y) coordinates of the points to be plotted, in the user's units. The PLOGzz subroutine will plot the logarithms of the numbers, where appropriate. The actual values within the coordinate arrays, however, will remain unaltered upon return from the MODE2 call of PLOGzz.

N Either real or integer; number of points to be plotted; sign as in MODE2 call of PLOT or PLOTM.

3) Plot Titles

```
     (PLOGX  )
     (PLOGXM )
     (PLOGY  )    *    *    *    *              *  *
CALL (PLOGYM )  (MODE,FORM,SIZE,DIR,XLOC,YLOC,FWA,N)
     (PLOGXY )
     (PLOGXYM)
```

or

```
     (PLOGX  )
     (PLOGXM )
     (PLOGY  )    *    *    *    *
CALL (PLOGYM )  (MODE,FORM,SIZE,DIR,XLOC,YLOC)
     (PLOGXY )
     (PLOGXYM)
```

MODE =3 (can be either real or integer).

FORM,SIZE DIR,FWA,N Either real or integer; same as in MODE3 call of PLOT or PLOTM.

XLOC,YLOC Real variables to specify the position, in the user's units, where the line of characters is to begin (lower left corner of the square enclosing the first character). The PLOGzz subroutines will begin the line of characters at the logarithms of XLOC and YLOC, where appropriate. The actual values of XLOC and YLOC, however, will remain unaltered upon return from the MODE3 call of PLOGzz.

4) Return Modified Parameters Passed in Last MODE1 Call of PLOGzz

```
     (PLOGX  )
     (PLOGXM )
     (PLOGY  )
CALL (PLOGYM )  (MODE,XLR,YLR,XMINR,XMAXR,YMINR,YMAXR,
     (PLOGXY )     TICXR,TICYR)
     (PLOGXYM)
```

MODE =4 (can be either real or integer).

The other arguments store the previous PLOGzz MODE1 arguments, modified as follows:

XMINR,XMAXR
YMINR,YMAXR — The axes maxima and minima, with logarithms taken where appropriate.

TICXR,TICYR — The tick value if axis linear and the tick code (as an integer) if axis logarithmic.

ROUTINES CALLED — PLOT and PLOTM, from AELIB. LOGPR12 and LOGPRn, $1 \leq n \leq 8$, utility routines loaded with the logarithmic plotting routines.

STORAGE REQUIRED — 400_8 for each of PLOGX,PLOGY and PLOGXY
1000_8 for utility routines.

EXIT — A normal exit returns control to the calling program. If errors are detected in the arguments passed, a diagnostic message is printed with traceback. If the error is fatal, the job is terminated. The possible errors are:

***MODE NOT 1, 2, 3, OR 4
First parameter invalid. Fatal error.

***INVALID LOG SCALE TIC VALUE SPECIFIED - DEFAULT IS 3
In a MODE1 call of PLOGzz, the LGTICz value was other than -1,0,1,2,3,4,5,11,12,13,14 or 15. Informative message. Execution continues with LGTICz set to the default value of 3.

****NO MODE1 CALL BEFORE MODE 2, 3 OR 4 CALL
A MODE1 call of PLOGzz must always precede a MODE 2, 3, or 4 call. Fatal error.

***ILLEGAL VALUES SPECIFIED FOR LOGARITHMIC SCALE
In a MODE1 call of PLOGzz, the MIN and MAX values for a logarithmic scale must both be greater than zero since the logarithms of negative numbers are not defined. Fatal error.

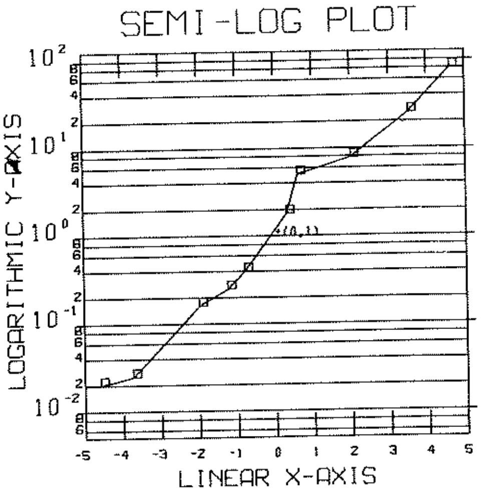
Since PLOT or PLOTM is called by PLOGzz, many of the PLOT diagnostic messages can be encountered as well.

EXAMPLE PROGRAM

The following program illustrates the use of the logarithmic plotting routine PLOGY. The routine AXLGY is used to plot centered axis titles and tic labels on the graph. (The plotted output has been reduced to half size for publication.)

```
      PROGRAM PLOGEX(INPUT=/8Ø,OUTPUT,PLOT)
      REAL X(1Ø),Y(1Ø),FORM(2)
      DATA N/1Ø/
      DATA X/-4.6,-3.75,-2.Ø,-1.25,-.8,.3,.6,2.Ø,3.5,4.6/
      DATA Y/.Ø2,.Ø25,.16,.25,.4,1.8,5.Ø,7.8,25.,8Ø./
      XL=4.Ø
      YL=4.Ø
      XMIN=-5.Ø
      XMAX=5.Ø
      YMIN=Ø.ØØ5
      YMAX=125.Ø
      ITICX=1
      LGTICY=4
C
C     PLOT A GRAPH WITH A LINEAR X-AXIS RUNNING FROM -5 TO 5 AND A
C     LOGARITHMIC Y-AXIS FROM .ØØ5 TO 125.
C     POSITION TICKS AT INTEGER VALUES ALONG X-AXIS AND DRAW LINES AT
C     (1,2,4,6,8)*1Ø**N ALONG Y-AXIS
C
      CALL PLOGY(1,XL,YL,XMIN,XMAX,YMIN,YMAX,ITICX,LGTICY)
C
C     PLOT CENTERED AXIS TITLES AND TICK LABELS ON THE GRAPH
C
      CALL AXLGY(17H("LINEAR X-AXIS"),13,22H("LOGARITHMIC Y-AXIS"),18,
     ,           4H(I2),3,1)
C
C     PLOT TITLE "SEMI-LOG PLOT" ABOVE GRAPH
C
      CALL PLOT(13,17H("SEMI-LOG PLOT"),3,Ø,Ø.44,4.2)
      PEN=1.Ø
      SYMB=2.Ø
C
C     PLOT DATA POINTS
C
      CALL PLOGY(2,PEN,SYMB,X,Y,N)
      FORM(1)=1ØH("+(Ø,1)"
      FORM(2)=1ØH)
      SIZE=1.Ø
      DIR=Ø.Ø
      XLOC=Ø.Ø
      YLOC=1.Ø
C
C     PLOT AND LABEL THE POINT (Ø,1)
C
      CALL PLOGY(3,FORM,SIZE,DIR,XLOC,YLOC)
      STOP
      END
```

PLOGX
PLOGXM
PLOGY
PLOGYM
PLOGXY
PLOGXYM



AUTHORS J.W. Wendorf and L.E. Evans DATE June 1975

TITLE

Plot a String of Text Using Two Character Sizes.

INTRODUCTION

This routine is similar to a PLOT or PLOTM call with MODE 3 or 13 except that a string of text can be plotted using two character sizes. The routine has two entry points:

VARSIZM used to label graphs if metric units have been used to plot the frame of the graph, i.e. a PLOTM MODE 1 call to plot the frame has been made before calling VARSIZM.

VARSIZ used to label graphs if English measure units have been used to plot the frame of the graph, i.e. a PLOT MODE 1 call to plot the frame has been made before calling VARSIZ.

ENTRIES

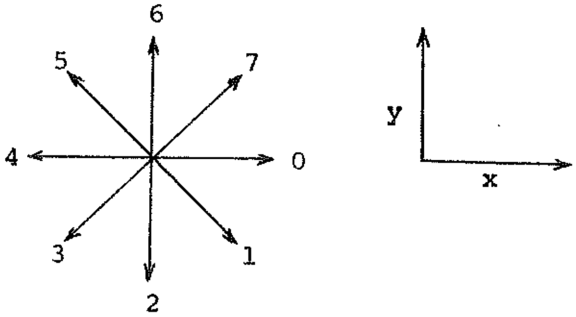
$\text{CALL}\left\{\begin{matrix}\text{VARSIZM}\\ \text{VARSIZ}\end{matrix}\right\}\text{(MODE,STRING,SIZE,DIR,XLOC,YLOC,CODE,LENGTH)}$

MODE — either real or integer input variable specifying whether XLOC,YLOC are expressed in user's units (MODE=3), or in inches (VARSIZ,MODE=13) or millimetres (VARSIZM, MODE=13).

STRING — an integer input variable or array containing a left justified Hollerith string. This Hollerith string contains the string of text to be plotted.

SIZE — either real or integer input variable specifying the larger of the two sizes of text to be plotted. The smaller size is equal to SIZE/2, rounded up if SIZE is odd, $1 \leq \text{SIZE} \leq 10$. Values outside this range will be replaced by SIZE=3. The size of the characters plotted is SIZE x 2 mm (.08") if DIR is even, and SIZE x $\sqrt{2}$ x 2 mm (.08") if DIR is odd.

DIR — either real or integer input variable specifying the direction of the character line, $0 \leq \text{DIR} < 7$. Values outside this range will be replaced by DIR=0. The angle the character line is plotted at is the same as with PLOT (i.e. $360^{\circ} - (\text{DIR} \times 45)^{\circ}$).



XLOC,YLOC real input variables specifying the position where the line of characters is to begin (lower left corner of the square enclosing the first character).

For MODE=3, XLOC, YLOC must be in user's units. For MODE=13, XLOC, YLOC must be in inches (VARSIZ), or millimetres (VARSIZM).

CODE an integer input variable or array containing a left justified Hollerith string specifying the size of the characters to be plotted from STRING. For each character in STRING including blanks there must be a corresponding letter in CODE to specify the size of test to be plotted. To plot the larger size of text an L is placed in CODE and for the smaller size an S is required.

LENGTH an integer input variable specifying the number of characters from STRING that are to be plotted, $L \geq 1$.

ROUTINES CALLED XIP and PLOT or PLOTM from AELIB.

EXIT If no errors are detected a normal exit returns control to the calling program. Any errors detected in the arguments passed to VARSIZ or VARSIZM result in the printing of a diagnostic message. The error conditions, associated messages and consequences are given below:

Fatal Errors (i.e. errors causing job termination):

*** ILLEGAL VALUE FOR MODE - MUST BE 3 OR 13
value for MODE invalid. MODE must be equal to 3 or 13. Job terminated.

*** ILLEGAL VALUE FOR LENGTH - MUST BE ≥ 1
Value for LENGTH invalid. LENGTH must be greater than or equal to 1. Job terminated.

*** ILLEGAL CODE CHARACTER
Illegal character in CODE. CODE characters must be either L or S. Job terminated.

Non-fatal Errors:

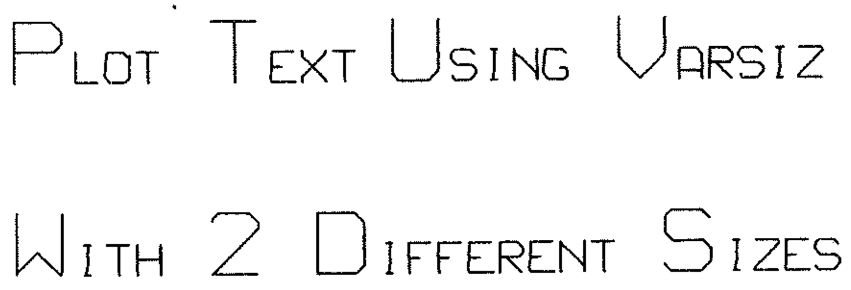
*** VALUE OUT OF RANGE FOR SIZE - DEFAULT IS 3
Value for SIZE invalid. Execution continues using a default value of SIZE=3.

*** VALUE OUT OF RANGE FOR DIRECTION - DEFAULT IS 0
Value for DIRECTION invalid. Execution continues using a default value of DIR=0.

PROGRAM EXAMPLE

The following program illustrates the use of VARSIZM by plotting a text string with the first letter of each word in "capitals".

```
      PROGRAM VARTEST(INPUT,OUTPUT,PLOT)
      INTEGER STRING1(3),STRING2(3),CODE1(3),CODE2(3)
      DATA STRING1/22HPLOT TEXT USING VARSIZ/
      DATA CODE1/22HLSSSSLSSSSLSSSSSLSSSSS/
      DATA STRING2/22HWITH 2 DIFFERENT SIZES/
      DATA CODE2/22HLSSSSLSLSSSSSSSSSLSSSS/
C
C     DEFINE PLOTTING SURFACE
C
      CALL PLOT(1,8.,8.,Ø.,8.,Ø.,8.,-1.,-1.)
C
C     INITIALIZE PARAMETERS FOR VARSIZ
C
      MODE=3
      SIZE=4.
      DIR=Ø.
      XLOC=1.
      YLOC=6.
      LENGTH=22
C
C     PLOT FIRST LINE OF TEXT
C
      CALL VARSIZ(MODE,STRING1,SIZE,DIR,XLOC,YLOC,CODE1,LENGTH)
C
C     PLOT SECOND LINE OF TEXT
C
      YLOC=5.
      CALL VARSIZ(MODE,STRING2,SIZE,DIR,XLOC,YLOC,CODE2,LENGTH)
      STOP
      END
```



AUTHORS P. Christie and L.E. Evans DATE July 1976

TITLE

Plot a Dashed Line Through a Set of Data Points with Optional Smoothing.

INTRODUCTION

Given a set of N data points, (X(I),Y(I)), I=1,...,N, this routine first defines a line (either a smooth curve or a line consisting of straight line segments joining the points)which passes through these points, and then plots this line using dashes. The dashed line is made up of repeated dash cycles, each cycle comprising four segments, a dash D1, a blank B1, a dash D2, and a blank B2. The dashed line is completely specified by a reference dash cycle length and four fractions providing the relative lengths of the segments in the dash cycle. These are the parameters DLENG and FRAC described below. There are two entry points to this routine, PLODAM (main entry) and PLODA. For PLODAM, all dimensioned quantities are in millimetres and for PLODA, all dimensioned quantities are in inches.

ENTRY

$\text{CALL}\left\{\begin{matrix}\text{PLODAM}\\ \text{PLODA}\end{matrix}\right\}$ (IOPT,X,Y,N,FRAC,DLENG,WS,LWS)

IOPT — integer input variable specifying the type of line desired to join the data points.

If IOPT=0, the data points are joined by straight line segments.

If IOPT=1, the data points are joined by a smooth curve (a cubic spline).

X(NX) — real input array of length $NX \geq N$ containing the abscissae of the data points to be plotted in the user's units. If IOPT=1, X must be strictly increasing, i.e. $X(I) \leq X(I+1)$.

Y(NY) — real input array of length $NY \geq N$ containing the ordinates (or function values) of the data points in the user's units.

N — integer input variable. $|N|$ is the number of data points and must be ≥ 2. N<0 specifies that the dashed line is to be drawn backwards through the data points, i.e. in the order (X(N),Y(N)),..., (X(1),Y(1)).

FRAC — real input array of dimension 4 to define relative lengths in the dash cycle.

DLENG real input variable. If DLENG≠0, then it specifies the reference length for the dash cycle in millimetres for PLODAM and in inches for PLODA. If DLENG=0, the reference length used is 1/4(average distance along the line between data points + minimum distance along the line between data points).

The parameter FRAC and the reference length together define the dashed line as follows:

FRAC(1)*reference length specifies length of segment D1.
FRAC(2)*reference length specifies length of segment B1.
FRAC(3)*reference length specifies length of segment D2.
FRAC(4)*reference length specifies length of segment B2.

$\sum_{i}$ FRAC(i)*reference length is then the actual length of the dashed cycle plotted.

WS(LWS) real input array of working storage required by PLODAM for IOPT=1. For IOPT=0, use a dummy variable for this argument.

LWS integer input variable specifying the length of WS as dimensioned in the calling routine. LWS$\geq$4*(N-1) for IOPT=1. For IOPT=0, use a dummy variable for this argument.

ROUTINES CALLED
ICSICU from IMSLIB
COSIMP from AELIB
FCNINT, COSAUX - utility routines provided with PLODAM.

COMMON BLOCKS USED
INTVL, DPARS

STORAGE REQUIRED
1000_8

EXIT

Normal Exit

If no errors are detected by PLODAM, control is returned to the calling routine.

(Note that input arrays X and Y are used internally by PLODAM to store the scaled coordinates for plotting. They then have been transformed back to contain the supplied user unit values. If high precision is required for subsequent processing, copies of these two arrays should be made before calling PLODAM.)

Error Exits

If an error is detected by PLODAM, the job will be terminated following the printing of an error message with traceback. The following is a list of error messages with error condition explanation (when necessary):

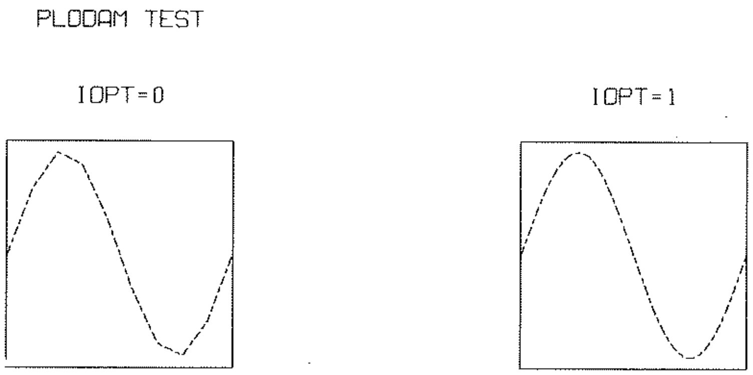
```
*** LESS THAN TWO INPUT DATA POINTS.
*** WORKING STORAGE PROVIDED IS NOT LONG ENOUGH.
     [LWS is not ≥ 4*N-1]
*** INPUT DATA NOT IN ASCENDING ORDER OF X.
```

PROGRAM EXAMPLE

The following program and plotted output illustrate the use of PLODAM to draw unsmoothed and smoothed dashed lines. (The plotted output has been reduced to half size for publication.)

```
      PROGRAM DAPLOT(INPUT,OUTPUT,PLOT)
      REAL X(1Ø),Y(1Ø),WS(4Ø),FRAC(4)
      REAL FORM(2)
C
C     INITIALIZATION OF PARAMETERS FOR PLOT FRAME AND THE DASH LINE.
C     WILL SPECIFY TO USE DEFAULT DLENG.
C
      DATA XL,YL,XMIN,XMAX,YMIN,YMAX,TICX,TICY,DLENG,FRAC/6Ø.,6Ø.,Ø.,6.3
     ,     ,-1.1,1.1,Ø.Ø,Ø.Ø,Ø.Ø,Ø.4,Ø.2,Ø.2,Ø.2/
C
C      SET UP THE ARRAYS SPECIFYING THE DATA POINTS,WHICH ARE
C     CHOSEN TO BE POINTS FROM A SINE CURVE FROM Ø. TO 2*PI
C
```

```
      N=1Ø
      DO 5Ø I=1,N
         X(I)=2.*3.1415926*(I-1)/(N-1)
         Y(I)=SIN(X(I))
 5Ø   CONTINUE
C
C     SET UP FRAME AND PLOT TITLE
C
      CALL PLOTM(1,XL,YL,XMIN,XMAX,YMIN,YMAX,TICX,TICY)
      SIZE=2.
      XLOC=8.Ø
      YLOC=9Ø.Ø
      DIR=Ø.Ø
      FORM(1)=1ØH("PLODAM T
      FORM(2)=1ØHEST")
      CALL PLOTM(13,FORM,SIZE,DIR,XLOC,YLOC)
C
C     PLOT AND LABEL CURVE USING STRAIGHT LINE SEGMENTS
C
      XLOC=18.Ø
      YLOC=7Ø.Ø
      FORM(1)=1ØH("IOPT=Ø")
      CALL PLOTM(13,FORM,SIZE,DIR,XLOC,YLOC)
      IOPT=Ø
      CALL PLODAM(IOPT,X,Y,N,FRAC,DLENG,WWS,LLWS)
C
C     SET UP FRAME, PLOT, AND LABEL CURVE USING CUBIC SPLINE
C
      CALL PLOTM(1,XL,YL,XMIN,XMAX,YMIN,YMAX,TICX,TICY)
      FORM(1)=1ØH("IOPT=1")
      CALL PLOTM(13,FORM,SIZE,DIR,XLOC,YLOC)
      LWS=4Ø
      IOPT=1
      CALL PLODAM(IOPT,X,Y,N,FRAC,DLENG,WS,LWS)
      STOP
      END
```



PROGRAMMED BY James Liu DATE June 1977

TITLE

Axis labelling subroutines:

AXLIN and AXLINM for both axes linear
AXLGX and AXLGXM for x axis logarithmic
AXLGY and AXLGYM for y axis logarithmic
AXLGXY and AXLGXYM for both axes logarithmic

ENTRIES

Each of these routines has two entry points, one for English units and one for metric units.
AXLIN,AXLGX,AYLGY and AXLGXY assume English units and AXLINM,AYLGXM,AXLGYM and AXLGXYM assume metric units.

$$\text{CALL}\begin{Bmatrix}\text{AXLIN}\\ \text{AXLINM}\end{Bmatrix}\text{(FORMX,NCX,FORMY,NCY,FORMXS,NCXS,FORMYS,NCYS)}$$

$$\text{CALL}\begin{Bmatrix}\text{AXLGX}\\ \text{AXLGXM}\end{Bmatrix}\text{(FORMX,NCX,FORMY,NCY,ITICXS,FORMYS,NCYS)}$$

$$\text{CALL}\begin{Bmatrix}\text{AXLGY}\\ \text{AXLGYM}\end{Bmatrix}\text{(FORMX,NCX,FORMY,NCY,FORMXS,NCXS,ITICYS)}$$

$$\text{CALL}\begin{Bmatrix}\text{AXLGXY}\\ \text{AXLGXYM}\end{Bmatrix}\text{(FORMX,NCX,FORMY,NCY,ITICXS,ITICYS)}$$

All of the arguments are listed below, grouped according to their functions and the axis to which they pertain.

Parameters for x-axis Title

FORMX(LFX) Real or integer input array containing a variable format for the x-axis title, for example,

```
FORMX(1)  = 10H(*X-AXIS T
FORMX(2)  = 10HITLE*)
```

NCX Integer input variable to specify the number of characters in the x-axis title. If zero, no title will be plotted and FORMX parameter will be ignored. Otherwise, this parameter is used to center the title parallel to and below the x-axis.

Parameters for y-axis Title

FORMY(LFY)
NCY
Analogous to FORMX,NCX, but for the y-axis title.

The title will be centered parallel to and to the left of the y-axis.

Parameters for Linear x-axis Tick Labels

FORMXS(LFXS)
Real or integer input array containing a variable format for the x-axis tick labels, for example,

FORMXS(1) = 10H(F3.1)

For AXLGX, AXLGXM, AXLGY, AXLGYM and AXLGXY, AXLGXYM, this may be a real or an integer format. (If the format contains the letter I, it is assumed to be an integer format. Otherwise it is assumed to be a real format.) If the TICX value used in the PLOT (or PLOTM) mode 1 or 11 call is not of the type specified by the format, then tick values are converted before being plotted.

For AXLIN and AXLINM, the format must be real.

Normally each tick along the x-axis will be numbered, but this may be reduced to every second tick to prevent overlap of labels.

NCXS
Integer input variable to specify the number of characters in the x-axis tick labels. If zero, no tick labels will be provided and FORMXS parameter will be ignored. Otherwise, this parameter is used to center the tick labels below the ticks.

Parameters for Logarithmic x-axis Tick Labels

ITICXS — Integer input variable, x-axis logarithmic tick label switch. If it equals zero, no x-axis tick labels will be printed. If it equals -1, only the powers of 10 will be labelled. Otherwise, each tick on the axis will be labelled using an appropriate format.

Parameters for Linear y-axis Tick Labels

FORMYS(LFYS)
NCYS — Analogous to FORMXS, NCXS, but for the y-axis tick labels.

The labels are printed to the left of the ticks.

Parameters for Logarithmic y-axis Tick Labels

ITICYS — Analogous to ITICXS, but for the y-axis tick labels.

ROUTINES CALLED — PLOT or PLOTM, from AELIB.
LOGPRn, for n=4,5 or 9 and LOGPRnn for n=10,11 or 12 are utility routines loaded with the axis labelling routines.

COMMON BLOCKS USED — AELERCM

STORAGE REQUIRED — 200_8 for each of AXLIN, AXLGX, AXLGY and AXLGXY.
1200_8 for utility routines.

EXIT — A normal exit returns control to the calling program. If errors are detected in the arguments passed, a diagnostic message will be printed with traceback. Some of these errors are fatal, causing the job to be terminated. Others are non-fatal, causing certain parts of the axis labelling to be abandoned but allowing the job to continue execution. The possible errors are:

$$***\left\{\begin{array}{l}\text{AXLGX}\\\text{AXLGY}\\\text{AXLGXY}\end{array}\right\}\text{ CALL NOT PRECEDED BY }\left\{\begin{array}{l}\text{PLOGX}\\\text{PLOGY}\\\text{PLOGXY}\end{array}\right\}\text{ MODE1 CALL}$$

AXLGzz makes a MODE4 call of PLOGzz in order to retrieve the scaling factors and tic values. Therefore, a MODE1 call of PLOGzz must precede a call to AXLGzz in the calling program. Fatal error.

***NEGATIVE VALUE SPECIFIED FOR NUMBER OF CHARACTERS - DEFAULT IS 0

A negative value was given for NCX, NCY, NCXS, NCYS. The value is assumed to be zero by the subroutine (which means the title or tick labels will not be plotted) and execution continues.

***NO MODE1 CALL BEFORE CALL OF MODE2,3,4 or 5

AXLIN(AXLINM) makes a MODE4 call of PLOT(PLOTM) in order to retrieve the scaling factors and tick values. Therefore, a MODE1 call of PLOT must precede a call to AXLIN in the calling program. Fatal error.

Note About Tick Labels:

The tick labels are printed in accordance with the scaling and ticking performed by the previous MODE1 call of PLOT or PLOGzz. If no ticking was done, no tick labels will be printed. If scaling was done in reverse order, i.e. XMIN was greater than XMAX, or YMIN was greater than YMAX, then the corresponding tick labels will be printed in descending order from left to right or bottom to top for the x-axis or y-axis respectively.

EXAMPLE

The use of the axis labelling routine AXLGY is illustrated in the writeup for the low level plotting routines (PLOGX,PLOGY,PLOGXY) and the use of AXLIN is shown in the writeup for PLOT.

AUTHOR

J. Wendorf and L.E. Evans DATE June 1975

TITLE Frame Definition and Axis Labelling for Semi-logarithmic Graphs:

SMLOGX and SMLOGXM for x axis logarithmic
SMLOGY and SMLOGYM for y axis logarithmic

ENTRIES Both routines have two entry points, one for English units and one for metric units. SMLOGX and SMLOGY assume English units. SMLOGXM and SMLOGYM assume metric units.

$$\text{CALL}\left\{\begin{matrix}\text{SMLOGX}\\ \text{SMLOGXM}\end{matrix}\right\}\text{(XL,YL,YMIN,YMAX,TIC,EXL,NC,FORMX,NTX,FORMY,}\\ \text{NTY,FORMSL,NCSL)}$$

$$\text{CALL}\left\{\begin{matrix}\text{SMLOGY}\\ \text{SMLOGYM}\end{matrix}\right\}\text{(XL,YL,XMIN,XMAX,TIC,EXL,NC,FORMX,NTX,FORMY,}\\ \text{NTY,FORMSL,NCSL)}$$

XL,YL real, graph dimensions in inches (for SMLOGX, SMLOGY) or mm (for SMLOGXM, SMLOGYM) in the x- and y-directions respectively.

XMIN,XMAX
YMIN,YMAX real, minimum and maximum values of the linear scale. The maximum value may be less than the minimum value.

TIC real, ticks are drawn at those points ±N * TIC (N=0,1,...), which fall within the interval (MIN,MAX) on the linear scale.

EXL real, largest exponent of 10 on the logarithmic scale (may be fractional). Note: If EXL is not fractional, NC cannot be zero.

NC integer, IABS(NC) is the number of complete cycles on the logarithmic scale. If NC>0, lines are drawn across the graph at 1×10^N, 2×10^N and 5×10^N for all appropriate N. If NC<0 ticks are drawn instead of lines.

FORMX format statement for the x-axis label, usually defined by a data statement in the calling program.

NTX integer, number of Hollerith characters in FORMX.

FORMY,NTY analogous to FORMX,NTX but for the y-axis label.

The above titles are plotted in size 2 (0.16" or 4 mm).

FORMSL format of the linear scale.

NCSL integer, number of characters (sign, decimal and digits) in the numbers of the linear scale. Ticks on the linear scale will be numbered XMIN (or YMIN) + N*TIC for appropriate N. The size is 1 (0.08" or 2 mm).

EXIT Control is returned to the calling program. No error checking is done.

ROUTINES CALLED PLOT or PLOTM from AELIB.

EXAMPLE The following program illustrates logarithmic plotting using SMLOGX. A graph is to be plotted which has a logarithmic X scale running from 1 x 10^{-1} to 5 x 10^{1}, and a linear y scale from -4 to 4. SMLOGX will be used to draw the plot frame, label the ticks and lines and print the centred axis titles. A MODE 13 call of PLOT will be used to position the title, "SEMI-LOG PLOT", above the graph. N data points, read in from cards will then be plotted using a MODE 2 call of PLOT.

```
PROGRAM SMLGEX(INPUT=/80,OUTPUT,PLOT)
REAL X(10),Y(10),FORMX(3),FORMY(2)
XL=4.0
YL=4.0
YMIN=-4.0
YMAX=4.0
TIC=1.0
EXL=ALOG10(50.0)
NC=2
FORMX(1)=10H(*LOGARITH
FORMX(2)=10HMIC X-AXIS
FORMX(3)=10H*)
NTX=18
FORMY(1)=10H(*LINEAR Y
      FORMY(2)=10H-AXIS*)
      NTY=13
      FORMSL=10H(F4.1)
      NCSL=4
      CALL SMLOGX(XL,YL,YMIN,YMAX,TIC,EXL,NC,FORMX,NTX,FORMY,
     $      NTY,FORMSL,NCSL)
      CALL PLOT(13,17H(*SEMI-LOG PLOT*),3,0,0.44,4.2)
      READ*,N
      READ*,(X(I),Y(I),I=1,N)
      DO 10 I=1,N
            X(I)=ALOG10(X(I))
   10 CONTINUE
      PEN=1.0
      SYMB=2.0
      CALL PLOT(2,PEN,SYMB,X,Y,N)
      STOP
      END
```

AUTHOR	J.F. Steljes	DATE	February 1972
	Revised by J.W. Wendorf		June 1975

TITLE — Frame Definition and Axis Labelling for Log-Log Graphs

ENTRIES — This routine has two entry points. LOGLOG assumes English units and LOGLOGM assumes metric units.

CALL$\left\{\begin{matrix}\text{LOGLOG}\\ \text{LOGLOGM}\end{matrix}\right\}$(XL,YL,EXXL,NCX,EXYL,NCY,FORMX,NTX,FORMY,NTY)

XL,YL — real, graph dimensions in inches (for LOGLOG) in mm (for LOGLOGM) in the x and y directions respectively.

EXXL — real, largest exponent of 10 on the x-axis scale (may be fractional).

NCX — integer, IABS(NCX) is the number of complete cycles on the x-axis scale. If NCS≥0, lines are drawn across the graph at 1 x 10^N, 2 x 10^N and 5 x 10^N for all appropriate N. If NCX<0 ticks are drawn instead of lines.

EXYL,NCY — analogous to EXXL,NCX for the y-axis scale.

FORMX — format statement for the x-axis label, usually defined by a data statement in the calling program.

NTX — integer, number of Hollerith characters in FORMX.

FORMY,NTY — analogous to FORMX,NTX but for the y-axis label.

The above titles are plotted in size 2 (0.16" or 4 mm).

EXIT — Control is returned to the control routine. No error checking is done.

ROUTINES CALLED — PLOT or PLOTM from AELIB.

STORAGE REQUIRED — 1100_8

AUTHOR — J.F. Steljes DATE February 1972

SIMPLT
SIMPLTM
DUMPON
DUMPOFF

TITLE

Simple Plotting Routine to Plot Linear, Semi-Log or Log-Log Graphs

INTRODUCTION

There are three logically separate functions performed by SIMPLT:

a) Replacement of defaults (titles, formats, scaling etc.) using data card input.
b) Frame setup including the plotting of axes, titles and axis labels.
c) Curve plotting with optional symbol key.

Combinations of these functions are invoked by the parameter JOBOPT described below. Functions b and c are required for each plot; a is optional.

There are four entry points to this routine. SIMPLT and SIMPLTM use English and metric units respectively; DUMPON and DUMPOFF control debugging printout.

ENTRIES

a) CALL $\left\{\begin{matrix}\text{SIMPLT}\\\text{SIMPLTM}\end{matrix}\right\}$ (JOBOPT,X,Y,N,PEN,SYMB,LOGOPT)

or

CALL $\left\{\begin{matrix}\text{SIMPLT}\\\text{SIMPLTM}\end{matrix}\right\}$ (JOBOPT,X,Y,N,PEN,SYMB)

JOBOPT integer input variable specifying the combination of functions to be performed as follows:

JOBOPT	Functions Performed
0	a, b and c
1	b and c
2	b and c without re-calculating maximums, minimums, tic values and tic label formats. In other words, a new graph will be plotted on a new frame but the axis limits will be the same as in the previous graph. The axis lengths, all titles and title locations, however, can be changed.

3	c only. Note that function b must have been performed at least once prior to this call. JOBOPT 3 plots data on an already existing axis frame.
4	a only.

X,Y real arrays containing the (X,Y) coordinates of the points to be plotted, in the user's units. If a logarithmic plot is being produced (see the LOGOPT option below), the logarithms of the appropriate coordinates will be used.

N either real or integer variable specifying the number of points to be plotted.

N>0, coordinate pairs are plotted in the order (X(1),Y(1)), (X(2),Y(2)), ..., (X(N),Y(N)).

N<0, coordinate pairs are plotted in the order (X(N),Y(N)), (X(N-1),Y(N-1)), ..., (X(1),Y(1)).

PEN either real or integer variable specifying pen control.

=0, pen is raised between points.
=1, pen is left on paper between points.

SYMB either real or integer variable specifying the symbol to be plotted at each point (as defined for PLOT).

LOGOPT optional integer variable specifying the type of graph to be plotted

LOGOPT	X-AXIS	Y-AXIS
0 or unspecified	linear	linear
1	logarithmic	linear
2	linear	logarithmic
3	logarithmic	logarithmic

b) CALL DUMPON

Following this call, optional parameters for SIMPLT will be printed out every time function b is executed.

c) CALL DUMPOFF

This call turns off the printing of optional SIMPLT parameters.

ROUTINES CALLED

ALERPR, TRACEB, PLOT,PLOTX,PLOGY,PLOGXY (or metric versions) and AXLIN,AXLGX,AXLGY,AXLGXY (or matric versions) from AELIB. LOGPRnn,nn = 13 to 21, utility routines loaded with SIMPLT.

COMMON BLOCKS USED

SCALES, FORMAT, SYMEAN and TITLOC

STORAGE REQUIRED

5300_8

EXIT

A normal exit returns control to the calling routine. Errors detected in the arguments passed to SIMPLT result in the printing of a diagnostic message with traceback. The error conditions and associated messages are listed below (all errors are fatal, causing job termination):

```
***JOBOPT NOT 0,1,2,3, OR 4
     First parameter in calling sequence is invalid.

***LOGOPT NOT 0,1,2 OR 3
     Last parameter in calling sequence is invalid.

***X AXIS DECLARED LT .005 OR GT 160 INCHES
     Parameter XL is illegal.

***Y AXIS DECLARED LT .005 OR GT 28.175 INCHES
     Parameter YL is illegal.
```

***ZERO VALUE FOR NUMBER OF X,Y PAIRS
Parameter N in calling sequence is zero.

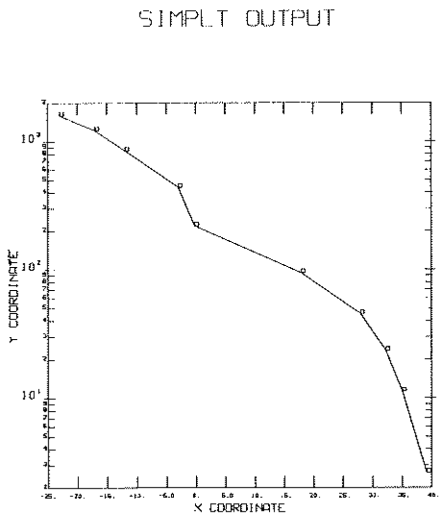
***ILLEGAL VALUES SPECIFIED FOR LOGARITHMIC SCALE
(a) If one of the parameters XMIN, XMAX, YMIN or YMAX refers to a logarithmic axis and is specified as ≤ 0; or (b) if a data value ≤ 0 is found by SIMPLT while scanning the coordinate array (X or Y parameter in calling sequence) of a logarithmic axis.

***JOBOPT 2 OR 3 CALL NOT PRECEDED BY JOBOPT 0 OR 1 CALL.
Section b of SIMPLT must have been executed at least once before a JOBOPT 2 or 3 call of SIMPLT is made.

PROGRAM EXAMPLE

This program produces a complete plot of a set of N data points. The graph has a linear x scale and a logarithmic y scale. The default size (8" x 8") plot has been reduced to one-quarter size for publication.

```
      PROGRAM SIMPLEX(INPUT,OUTPUT,PLOT)
      REAL X(1Ø),Y(1Ø)
      DATA X/-23.Ø, -17.Ø, -12.Ø, -3.Ø, -Ø.2, 18.Ø, 28.Ø, 32.3, 35.1,
     ,39.2/
      DATA Y/1587.6, 1217.3, 851.5, 437.2, 218.Ø, 93.6, 45.1, 23.3,
     ,11.2, 2.6/
      N=1Ø
      JOBOPT=1
      PEN=1
      SYMB=2
      LOGOPT=2
      CALL SIMPLT(JOBOPT,X,Y,N,PEN,SYMB,LOGOPT)
      STOP
      END
```



***** END OF BASIC CALLING INFORMATION FOR SIMPLT *****

Additional Options for SIMPLT

To change the appearance of the default plot produced by SIMPLT, additional parameters can be specified by

i) supplying input data cards and using JOBOPT=0 or 4, or by

ii) defining entries in labelled common blocks without changing JOBOPT.

The list of such parameters (with defaults) and an example program defining some of them are provided following a brief description of the two modes of input.

i) Data Card Input

If JOBOPT has a value of 0 or 4, SIMPLT will read data cards from the input file. Any or all of the optional parameters can be defined, free-format, as follows:

```
                          format=(string),END
          ARRAYNAME(n)=value,m*value,value,...,value,
          VARIABLE=value,...,ARRAYNAME=value,m*value,..
```

where

variable — is the name of any parameter that is a simple variable, for example: XMIN,TICY,TITLOCX.

arrayname — is the name of any parameter that is an array (excluding those meant to hold a variable format); values are to be assigned starting at the beginning of the array; for example: STVARS,SYMCOD.

arrayname(n) — is used if values are to be assigned to the array starting at the nth entry.

format — is the name of any parameter meant to hold a variable format; for example: FTITLE,FXSCAL,FSYMB1. (Note that FXSCAL and FYSCAL are considered "variable" rather than "format" when they refer to a logarithmic axis.)

value — is the value to be assigned to the SIMPLT parameter.

m*value — is used if that value is to be repeated m times

(string) — is a format for a variable format parameter

END — signals the end of data input. It is the only required input parameter.

Data items must be separated by commas. All 80 columns of a data card may be used and data items may be continued onto column 1 of the next card.

Each time data input is requested, the next data card is read and SIMPLT will continue to read cards and assign values to parameters until "END" is detected.

If errors are encountered during this data input, the offending data card is printed with a pointer to the last good column, a diagnostic message is issued, and the job is terminated.

ii) Common Block Layout for Additional Parameters

The additional parameters are stored by SIMPLT in four labelled common blocks defined as follows:

```
COMMON/SCALES/XMIN,XMAX,TICX,FXSCAL,YMIN,YMAX,TICY,FYSCAL
COMMON/FORMAT/XL,YL,FTITLE(8),FSUBTIT(8),FXAXTIT(8),FYAXTIT(8),
     NSTVARS,STVARS(10)
COMMON/SYMEAN/SYMCOD(10),FSYMB1(2),FSYMB2(2),FSYMB3(2),FSYMB4(2),
     FSYMB5(2),FSYMB6(2),FSYMB7(2),FSYMB8(2),FSYMB9(2),FSYMB10(2),
     SYMLOCX,SYMLOCY
COMMON/TITLOC/TITLOCX,TITLOCY,SUBLOCX,SUBLOCY,XAXLOCX,XAXLOCY,
     YAXLOCX,YAXLOCY
```

So, to change the default value for any of these parameters, define the appropriate common block to its full length, and then assign a value to the desired parameter(s)(use assignment statements. not DATA statements). Parameters mentioned in a common block but not changed by the user will retain their default values.

Once a parameter has been defined by the user, it retains that definition until changed once again by the user. Many of the parameters have "action defaults", i.e. the default requires SIMPLT to perform some kind of calculation. This is true of all the "scales parameters" (the elements of common block /SCALES/), and of the "location parameters" (SYMLOCX, SYMLOCY and the elements of common block /TITLOC/). If one of these "action defaults" has been changed by the user, i.e. the user has defined the parameter, SIMPLT will no longer perform the associated calculation, but rather, will use the defined value. If the user should later wish to return that parameter to its "action default", this can be done by setting that parameter to the octal constant "37775555555555555555".

List of Additional Parameters for SIMPLT

XMIN,XMAX
YMIN,YMAX — real bounds on variable values. If XMIN>XMAX(YMIN>YMAX) the plot will be scaled in descending order from left to right (bottom to top). Default: SIMPLT computes values.

TICX,TICY — either real or integer variables, the tick values to be used in the PLOT, PLOGX, PLOGY or PLOGXY MODE1 call. Default: SIMPLT computes values.

FXSCAL,FYSCAL — For a linear axis; either real or integer variable containing a variable format for the axis tick labels; for example, FXSCAL=10H(F3.1).

If the format contains the letter "I", the tick label numbers plotted will be integers. Otherwise, they will be real numbers.

For a logarithmic axis; integer variable controlling the logarithmic tick labelling. If it equals zero, no tick labels will be printed on that axis. If it equals -1, only the powers of 10 will be labelled. Otherwise, each tick on the axis will be labelled using an appropriate format.
Default: SIMPLT computes values appropriate to the data.

XL,YL

Real variables specifying the lengths of the axes in in. for SIMPLT and in mm for SIMPLTM.

$.005 < XL \leq 160.0$	for	$.127 \leq XL \leq 4064.$	for
$.005 \leq YL \leq 28.175$	SIMPLT	$.127 \leq YL \leq 715.645$	SIMPLTM

Default values: XL=YL=8.0 for SIMPLT
XL=YL=203.2 for SIMPLTM

FTITLE

Real or integer array 8 words long, containing a variable format for the plot title. The format must consist of a list of format specifications enclosed in parentheses, but without the word FORMAT or the statement label.
Default: (*SIMPLT OUTPUT*)

FXAXTIT,
FYAXTIT

Real or integer arrays of length 8 words each, containing variable formats for the x-axis and y-axis titles, respectively. Each format must consist of a list of format specifications enclosed in parentheses, but without the word FORMAT or the statement label.
Default: (*X COORDINATE*) and (*Y COORDINATE*).

FSUBTIT

Real or integer array of length 8 words, containing a variable format for the plot subtitle, for example: (≠PLOT NUMBER≠,I3,20X,≠DATE:≠3F2.0). The format must consist of a list of format specifications enclosed in parentheses, but without the word FORMAT or the statement label.
Default: (*   *)

STVARS

Real or integer array of length 10 words, containing the list of variables to be plotted under the format passed in FSUBTIT. The following data could be used with the above format:
DATA STVARS/1,30.,7.,75./
Default: Entire array is left undefined.

NSTVARS

Integer variable, the number of variables to be output in the subtitle. In the example above, NSTVARS should equal 4. If no data is required by FSUBTIT, NSTVARS must be 0.
Default value: NSTVARS=0.

SYMCOD Either real or integer array of length 10 words, supplying the symbol codes to be included in the symbol key.
Default values: The entire array is preset to -1.0 (-1.0 does not correspond to any symbol).

FSYMB1,FSYMB2
FSYMB3,FSYMB4
FSYMB5,FSYMB6
FSYMB7,FSYMB8
FSYMB9,FSYMB10 Either real or integer arrays of length two words each, containing variable formats for the meanings corresponding to the symbols defined in the SYMCOD array. FSYMB1 corresponds to SYMCOD(1),FSYMB2 to FSYMCOD(2), and so on. For example, if SYMCOD(1)=2 and FSYMB1 contains (≠EXPERIMENTAL≠), then "   EXPERIMENTAL" will be plotted whenever a set of points is plotted using the symbol " " (SYMB=2).
Default: All of the arrays are preset to (*   *).

SYMLOCX,
SYMLOCY Real variables specifying the position, in in., for SIMPLT and in mm for SIMPLTM, of the lower left corner of the first symbol of the first symbol key entry to be plotted.
Default: SIMPLT places symbol key above the plot frame with entries one beneath the other. If not enough space is available, SIMPLT will plot some key entries to the right of those already plotted.

SUBLOCX,
SUBLOCY Real variables specifying the position of the plot subtitle in in. for SIMPLT and in mm for SIMPLTM.
Default: SIMPLT centers the subtitle above the plot below the plot title.

XAXLOCX,
XAXLOCY Real variables specifying the position of the x-axis title in in. for SIMPLT and in mm for SIMPLTM.
Default: SIMPLT centers the x-axis title below the x-axis.

YAXLOCX,
YAXLOCY Real variables specifying the position of the y-axis title in in. for SIMPLT and in mm for SIMPLTM.
Default: SIMPLT centers the y-axis title to the left of the y-axis.

TITLOCX,
TITLOCY Real variables specifying the position, in in. for SIMPLT and in mm for SIMPLTM, of the lower left corner of the first character of the plot title.
Default: SIMPLT centers the title above the plot frame.

PROGRAM EXAMPLE

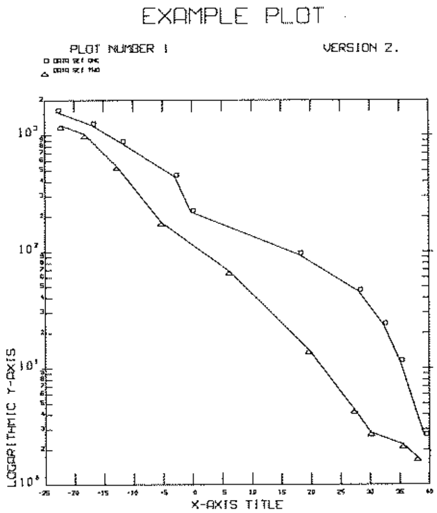
This program has changed the defaults of the earlier example by:

- defining plot and axis titles
- adding a subtitle
- setting minimum value on logarithmic scale to a power of 10.
- specifying the location of the y-axis title.
- specifying the x-axis tick labels to be integers.
- plotting two sets of data and specifying symbol meanings.

The default size (8" x 8") plot has been reduced to one-quarter size for publication.

```
      PROGRAM SIMPLX2(INPUT,OUTPUT,PLOT)
      REAL X(10),Y(10),X2(10),Y2(10)
      COMMON/SCALES/XMIN,XMAX,TICX,FXSCAL,YMIN,YMAX,TICY,FYSCAL
      COMMON/FORMAT/XL,YL,FTITLE(8),FSUBTIT(8),FXAXTIT(8),FYAXTIT(8),
     ,NSTVARS,STVARS(10)
      COMMON/SYMEAN/SYMCOD(10),FSYMB1(2),FSYMB2(2),FSYMB3(2),FSYMB4(2),
     ,FSYMB5(2),FSYMB6(2),FSYMB7(2),FSYMB8(2),FSYMB9(2),FSYMB10(2),
     ,SYMLOCX,SYMLOCY
      COMMON/TITLOC/TITLOCX,TITLOCY,SUBLOCX,SUBLOCY,XAXLOCX,XAXLOCY,
     ,YAXLOCX,YAXLOCY
      DATA X/-23.0, -17.0, -12.0, -3.0, -0.2, 18.0, 28.0, 32.3, 35.1,
     ,39.2/
      DATA Y/1587.6, 1217.3, 851.5, 437.2, 218.0, 93.6, 45.1, 23.3,
     ,11.2, 2.6/
      DATA X2/-22.3, -18.2, -12.8, -5.3, 6.1, 19.6, 27.3, 30.1, 35.5,
     ,38.0/
      DATA Y2/1221.2, 1013.7, 538.3, 178.8, 67.7, 14.3, 4.4, 2.8, 2.2,
     ,1.7/
      N=10
      YMIN=1.0
      FXAXTIT(1)=10H(*X-AXIS T
      FXAXTIT(2)=10HITLE*)
      SYMCOD(1)=SYMB1=2.0
      SYMCOD(2)=SYMB2 =4.0
      FSYMB1(1)=10H(*DATA SET
      FSYMB1(2)=10H ONE*)
      FSYMB2(1)=10H(*DATA SET
      FSYMB2(2)=10H TWO*)
      YAXLOCY=0.0
      JOBOPT=0
      PEN=1
      LOGOPT=2
      CALL SIMPLT(JOBOPT,X,Y,N,PEN,SYMB1,LOGOPT)
      JOBOPT=3
      CALL SIMPLT(JOBOPT,X2,Y2,N ,PEN,SYMB2,LOGOPT)
      STOP
      END
```

```
FXSCAL=(I3), FTITLE=(*EXAMPLE PLOT*),
 FSUBTIT=(*PLOT NUMBER* ,I2,20X,*VERSION*,F3.0),
NSTVARS=2, STVARS=1,2.0,FYAXTIT=(*LOGARITHMIC Y-AXIS*), END
```



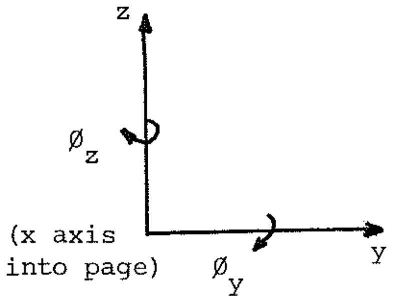
AUTHOR J.W. Wendorf DATE August 1975

DOCUMENTATION
REVISED BY L.E. Evans April 1978

TITLE Three Dimensional Perspective Plotting for a Surface

INTRODUCTION Given x and y arrays defining a rectangular x-y grid, a surface is defined by providing a value for each grid point, i.e. z(i,j)=height of the surface at grid point (x(i),y(j)). The only scaling carried out by SPACE is that required to fit the final 2-dimensional projection onto the plotting surface. Therefore, it is recommended that the values supplied in the x, y and z arrays should be scaled by the user to be the same order of magnitude before calling SPACE.

The orientation of the surface plotted is defined by the angles $\emptyset_y$ and $\emptyset_z$. The x axis goes into the page and $\emptyset_y$ and $\emptyset_z$ are defined as follows:



$0 \le \emptyset_z \le 90^\circ$

$-90^\circ < \emptyset_y < 90^\circ$

To see all three dimensions, $\emptyset_z$ should be non zero. $\emptyset_y > 0$ looks at the surface from the top. $\emptyset_y < 0$ looks at it from underneath.

Perspective is obtained by viewing the surface from a distance proportional to the viewing factor, VU. For large VU (i.e. $VU \sim 100$), the perspective plot approaches a parallel projection.

ENTRIES This subroutine has two entry points: SPACE assumes English units and SPACEM, metric units.

$\text{CALL}\left\{\begin{matrix}\text{SPACE}\\ \text{SPACEM}\end{matrix}\right\}$ (X,Y,Z,NX,NY,NZ,PHIY,PHIZ,VU,WIDTH)

X,Y real input monotonically increasing arrays defining on x-y grid.

Z(NRZ,NCZ) real input array of function values, Z(I,J) is the height of the surface at grid point (X(I),Y(J)). $NRZ \ge NX$, $NCZ \ge NY$.

NX integer input variable, the number of elements in X.

NY — integer input variable, the number of elements in Y, $NY \le 64$.

NZ — integer input variable, the number of rows of Z as dimensioned in the calling program.

PHIY — real input variable, the number of degrees the surface is to be rotated about the y axis, $|\phi_y| < 90^\circ$.

PHIZ — real input variable, the number of degrees the surface is to be rotated about the z axis, $0 \le \phi_z \le 90^\circ$.

VU — real input viewing factor controlling the distance of the viewing point from the surface, $VU \ge 1$.

WIDTH — real input variable. If WIDTH<0, the 3D frame will be drawn with the surface. If WIDTH>0 only the surface is drawn.
For SPACE, the plotting area will be $|WIDTH|$ in. wide, $|WIDTH| \le 10.0$
For SPACEM, the plotting area will be $|WIDTH|$ mm wide, $|WIDTH| \le 254$.

(Wider plots are possible with a special version of SPACE. Contact plotting analyst for details.)

ROUTINES CALLED — PRESET, PLOT and PLOTM from AELIB.

STORAGE REQUIRED — 7200_8.

EXITS — <u>Normal Exit</u>

If no error conditions are detected, the plot is completed and control is returned to the calling routine. SPACE(SPACEM) plots only the surface and optionally the frame. All titles and labels must be supplied by the user. Scaling information for this can be retrieved by a MODE4 call of PLOT(PLOTM).

Error Exits

The message

SPACE PARAMETER ERROR NUMBER N

indicates that SPACE was unable to complete the plot as requested. The error conditions corresponding to values of N are as follows:

-2 $\emptyset_y$ too large for viewpoint; viewpoint adjusted and plot finished

-1 $\emptyset_z$ too large for viewpoint; viewpoint adjusted and plot finished

1 X or Y array not monotonically increasing or Z values constant; no plotting done

2 $\emptyset_y$ and/or $\emptyset_z$ outside allowable range; no plotting done

3 NY too large; no plotting done

4 WIDTH outside limits; no plotting done

5,6 overflow of internal working storage (see plotting analyst)

7 viewing factor less than 1.0; a value of 1.0 is used

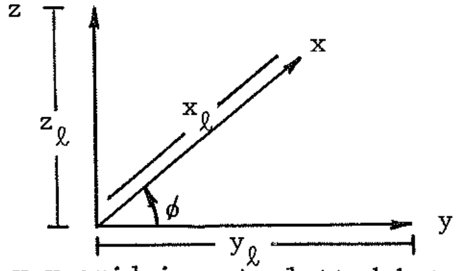
10 plotting surface improperly defined (see plotting analyst); no plotting done.

AUTHOR E. Myles and G. Sutherland DATE June 1973

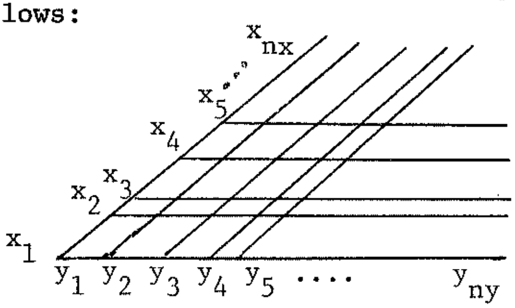
REVISED L. Evans March 1978
L. Evans May 1981

TITLE Surface, Plotting by Parallel Projection

INTRODUCTION Given x and y arrays defining a rectangular x-y grid, a surface is defined by providing a value for each grid point, i.e. z(i,j) = height of the surface at grid point (x(i),y(j)). The scale and orientation of the parallel projection of this surface are defined by the three projected axis lengths x_ℓ, y_ℓ and z_ℓ and the angle, ϕ, between the projected x and y axes are as shown below:



The x-y grid is not plotted but is implicitly projected as follows:



ENTRIES This routine has two entry points, PLOT3D for English units and PLOT3DM for metric units.

$\text{CALL}\left\{\begin{matrix}\text{PLOT3D}\\ \text{PLOT3DM}\end{matrix}\right\}$ (XL,YL,ZL,X,Y,Z,NX,NY,NRZ,PHI)

XL,YL,ZL real input variables, the lengths of the projected x,y and z axes (resp.) in inches (for PLOT3D) and in millimetres (for PLOT3DM)
YL+YL*cos $\phi \leq$ 20 in (508 mm)
ZL $\leq$ 10 in (254 mm)

X,Y — real input arrays specifying the grid points on which the surface is defined. Both arrays must be monotonically increasing.

Z(NRZ,NCZ) — real input array specifying the height of the surface at each grid point. Z(I,J) is the height of the surface at grid point (X(I),Y(J)). Z values must be positive and the minimum value is assumed to be zero.

NX,NY — integer input variables, the number of elements in X and Y resp. $NY \leq 64$.

NRZ — integer input variable, the number of rows dimensioned for Z in the calling program.

PHI — real input variable, the angle between the projected y and x axes. $PHI < 90^{o}$.

ROUTINES CALLED — PLOT or PLOTM from AELIB.

STORAGE REQUIRED — 7000_8

EXITS — Normal Exit

The plotting area is defined and the surface is plotted before returning, control to the calling routine. (Only the surface is plotted by PLOT3D(PLOT3DM). All titles and labels must be done by subsequent plotting calls. The plotting frame defined by PLOT3D(PLOT3DM) is

$y_\ell+x_\ell \cos \phi$ long by 10 in (254 mm) wide

and the user units are

0 to 100 ($y_\ell+x_\ell \cos \phi$) in the x plotting direction

and

0 to 1000 in the y plotting direction. Thus the user unit coordinates of any point (x',y') on the projected plot are the distances from the plotting origin expressed in hundredths of an inch.)

Error Exit

If $y_\ell + x_\ell \cos \phi > 20$ in (508 mm) the message

PLOT LENGTH = ______ EXCEEDS 20 IN.

is printed and the program is terminated. No plotting is done.

AUTHOR	D. McPherson	DATE	October 1969
DOCUMENTATION REVISED	L. Evans		March 1978

TITLE Contour Plotting Routine

INTRODUCTION The plotting surface on which the contours are to be drawn must be set up first by a call of PLOT or PLOTM with MODE set to 1. The two entry points to the contour plotting routine are CONTOUR and CONTOUM. CONTOUR (CONTOUM) must be used if the frame was established by PLOT(PLOTM) even though no English or metric units are explicitly used by this routine.

ENTRIES CALL$\left\{\begin{matrix}\text{CONTOUR}\\ \text{CONTOUM}\end{matrix}\right\}$ (X,Y,Z,NX,NY,NRZ,START,STEP,FIN)

X,Y real input arrays together specifying a grid of X and Y values. X and Y should be monotonically increasing. (Note that X and Y may define unequally spaced grid points.)

Z(NRZ,NCZ) real input array of Z values defined on the X-Y grid. Z(I,J) is the height of the surface at grid point (Y(I),X(J)). NRZ$\geq$NY, NCZ$\geq$NX. Because of working storage space limitations within CONTOUR, NX*NY must be $\leq$ 8400.

NX,NY integer input variables, the number of entries in the X and Y arrays (respectively)

NRZ integer input variable, the number of rows of Z as dimensioned in the calling program

START real input variable specifying the Z value for the first contour to be drawn

STEP real input variable specifying the Z value increment at which contours are to be drawn

FIN real input variable specifying the maximum Z value at which a contour is to be drawn.

Note: If only one contour is to be drawn, set START to be the value of that contour and set both FIN and STEP to a value slightly larger than START.

EXIT (a) If too many data points are supplied to CONTOUR, i.e. NX*NY > 8400, the message

```
***MORE THAN 8400 GRID POINTS FOR CONTOUR PLOT,
NX=      ,NY=
```

is printed with traceback and execution is terminated.

(b) Otherwise, contours of constant Z are drawn for Z=START+N*STEP for N=0,1,2..... If a contour value is equal to a surface value at a border grid point, the routine may not be able to draw that contour. In this case the message

```
***THE CONTOUR VALUE ___ IS EQUAL TO Z(__,__).
THAT CONTOUR MAY BE MISSED.
```

is printed with traceback and execution continues.

Contours of value Z=START+N*STEP are attempted for all values of N such that Z≤FIN. Control is then returned to the calling routine.

ROUTINES CALLED — PLOT, PLOTM and ALERPR from AELIB.

STORAGE REQUIRED — 2300_8

EXAMPLE — The following program

- defines a 50x20 x-y grid
- reads in Z values, one for each grid point
- sets up the 10 in. x 5 in. plotting frame, and
- plots a contour map of the surface.

```
      PROGRAM LAND(INPUT,OUTPUT,PLOT)
C
C        TEST PROGRAM TO ILLUSTRATE USE OF CONTOUR
C
      DIMENSION X(50),Y(20),Z(20,50)
C
C        INITIAL VALUES FOR CALL TO CONTOUR
C
        DATA NX,NY,NZ/50,20,20/
       STEP=10.
      START=70.
      FIN=220.
C
C        SET UP X AND Y ARRAYS
C
      DELX=600./49.
      DO 10 I=1,50
   10 X(I)=FLOAT(I-1)*DELX
C
      DELY=300./19.
      DO 20 I=1,20
   20 Y(I)=FLOAT(I-1)*DELY
```

```
C
C     READ IN Z THE TWO-DIMENSIONAL ARRAY
C
      DO 30 I=1,20
   30 READ 31,(Z(I,J),J=1,50)
C
C     MODE 1 CALL TO PLOT TO SET UP THE PLOTTING SURFACE FOR CONTOUR
C
      CALL PLOT(1,10.,5.,0.,600.,0.,300.,0,0)
C
      CALL CONTOUR(X,Y,Z,NX,NY,NZ,START,STEP,FIN)
C
   31 FORMAT(20F4.0)
      END
```

AUTHOR	Translated from Apex by P.C. Barnett	DATE	November 1969
REVISED BY	D.G. Stewart and L.E. Evans	DATE	March 1978

TITLE Printer Plot Routine

ENTRY CALL PROT(NV,NI,NT,X,Y,W,Z)

NV integer input variable, the number of dependent variables to be plotted on a single set of axes (up to 3).

NI integer input variable, the number of points per curve (up to 100).

NT integer input variable, if NT=1, data cards with titles are to be read in. If NT=0, no data cards will be read and the titles printed will be those currently stored in the common block TITLES, defined below. These titles retain default values, set at load time, unless they are changed by reading of titles on a previous call to PROT or by assignment of values in the calling program.

X real input array of length $\geq$ NI, containing the values of the independent variable arranged in ascending order.

Y,W,Z real input arrays containing the values of the dependent variables. Each array has length $\geq$ NI.

Thus the call may have 5, 6 or 7 parameters. For example:

```
CALL PROT(1,25,0,A,B)
CALL PROT(3,50,1,D,E,F,G)
```

ADDITIONAL ENTRY INFO If NT=1, the user must supply three data cards as follows:

CARD 1 Title of the plot in 8A10 format.

CARD 2 Title for the independent variable in 10A1 format.

CARD 3 Titles for the dependent variables in 3A10 format (up to three titles of 10 characters each).

These are used for labelling the plot and the axes.

COMMON BLOCKS USED

(1) COMMON/TITLES/TITLE(8),XTITLE(10),YTITLE(3)

This common block may be used to pass titles to PROT where

TITLE(8) contains the plot title in 8A10 format.

XTITLE(10) contains 10 characters for the independent variable (i.e. vertical) label with the ith character left justified in XTITLE(i).

YTITLE(3) contains up to three dependent variable labels each in A10 format.

(2) COMMON/PRCNT/IPF,I

This common block may be used to direct the printer plot to a file other than OUTPUT. IPF is an integer variable, set by default to 0 by PROT. If IPF is set to 1 by the calling program, the printer plots are sent to logical unit I, i.e. TAPEI, instead of to OUTPUT.

ROUTINES CALLED

PRESET from AELIB.

STORAGE REQUIRED

2100_8

EXIT

The independent and dependent variable values are scaled and then the curves are plotted. The first curve is plotted using *'s, the second using +'s and the third using Λ's. If two or more curves coincide, :'s are plotted for the coincident points.

The plot title and symbol key are printed along with dependent variable (i.e. horizontal) axis labels above the plot and independent variable (i.e. vertical) axis labels are printed to the left of the plot. Control is returned to the calling program after the plot has been generated. No error checking is performed.

SPEED	The speed depends on the number of dependent variables and the number of points to be plotted. For two dependent variables, 50 points each, the time is approximately 50 ms (170 Model 175, October 1978).

AUTHOR	S.L. Likeness	DATE	June 1971
REVISED BY	D.G. Stewart	DATE	September 1978

TITLE Position Plotter Pen on Graph Paper

INTRODUCTION To plot on graph paper (or any other special form) the operators must be able to set the plotter pen to the correct initial position on the paper and to check it before proceeding with the plot. If SPLOT is called before the first PLOT/PLOTM mode 1 call, such set up is possible.

A call to SPLOT causes the job name to be plotted, then the plotter waits until a 3300 console reply is given. The operator will respond when the pen has been positioned properly. A draftsman's L is plotted for checking before the first frame is plotted. (If multiple copies of a plot are to be produced, the above procedure is repeated for each copy.)

For more information about the use of special plotting forms, see the informal document, Special Computer Output, available from the Mathematics and Computation Branch Office.

ENTRY CALL SPLOT

EXIT Control is returned to the calling routine.

AUTHOR J.F. Steljes DATE April 1972

TITLE

Define plotting surface and initialize parameters for Calcomp plotting routines. This routine must be the first Calcomp routine called by a user's program.

INTRODUCTION

The original name for this routine was PLOTI, and this entry point is being retained to allow existing programs to run.

PLOTCI is now the recommended entry point which was introduced to take advantage of new features in the basic graphics subroutines which support the Calcomp routines. Users replacing their existing calls to PLOTI with calls of PLOTCI will notice a shift in the origin of their plots by two inches to the left and half an inch down.

If the Calcomp routines are being used by an overlay program, PLOTCI should reside in the (0,0) overlay to ensure that the common blocks it uses are initialized only once.

ENTRY

```
CALL PLOTCI(XL,YL)
CALL PLOTI(XL,YL)
```

where arguments are defined as follows:

XL real, input variable specifying the length in inches of plotting surface required where $.005 \leq XL \leq 163.0$. The plotting surface, i.e. paper received, will be this long regardless of the size of the actual plot produced.

YL real, input variable specifying the width in inches of surface required where $.005 \leq YL \leq 28.675$. If YL is less than or equal to 10.675, the plot will be produced on narrow paper. If greater than 10.675, it will automatically be produced on wide paper.

ROUTINES CALLED

PLOTC from AELIB.

COMMON BLOCKS USED

CALCUR, CALCNT, CALBUF

EXITS

Normal Exit:

A new plotting frame is initialized via a call to PLOT with MODE=10. (For PLOTI, a MODE=1 call is made.) In addition, the current pen position and plotting origin are set to (0.,0.), and the plotting factor is set to 1. before control is returned to the calling routine.

Error Exit:

If either XL or YL is outside the allowed range or illegal (i.e. indefinite or infinite) the job is terminated by PLOT after an error message has been issued.

EXAMPLE

See program example in writeup of the AELIB routine PLT or LINE for an illustration of the use of PLOTCI.

AUTHOR	L.E. Evans	DATE	November 1975
REVISED	G.N. Williams		March 1980

TITLE

Calcomp routine to set the plotting factor; i.e. the ratio of the desired plot size to the normal plot size, for Calcomp plotting routines.

ENTRY

CALL FACTOR(FACT)

FACT real input variable specifying the plotting factor required. Subsequent plotting from Calcomp routines up to the next PLOTCI or FACTOR call is to be produced at FACT times the normal plotting size. It should be noted that:

(a) The allowable size of plotting surface is not affected by this call so it is up to the user to ensure that the scaled plot will fit in the plotting surface defined.

(b) The scaling of the plot may not be exact for characters or special symbols because only a discrete range of sizes is allowed, as specified in the writeup of PLOT.

COMMON BLOCKS USED

CALCUR, CALCNT, CALBUF

EXIT

The plotting factor is set to the value passed in FACT and control is returned to the calling routine.

EXAMPLE

Assuming that the PLOTCI and FACTOR calls shown are the only calls to these routines in the sequence below, then all plotting from Calcomp routines between statements 5 and 10 will be at the normal size, plotting between statements 10 and 20 will be at half size, and plotting after statement 20 will be at two and one half times normal size.

```
      XM=5.
      YM=5.
    5 CALL PLOTCI(XM,YM)
      :
      FACT=.5
   10 CALL FACTOR(FACT)
      :
      FACT=2.5
   20 CALL FACTOR(FACT)
      :
```

AUTHOR L.E. Evans DATE January 1976

TITLE

Calcomp routine to retrieve (a) the current factor and (b) the current pen coordinates.

ENTRY

CALL WHERE(RX,RY,RFACT)

RX,RY real output variables returning the pen coordinates set by the previous Calcomp call which changed the pen position.

(a) If preceded by a call to PLT or to SYMBOL using special symbols, then (RX,RY) are the coordinates of the last point or symbol referenced.

(b) If preceded by a call to NUMBER or to SYMBOL using characters, then (RX,RY) are the coordinates of the lower left-hand corner of the next character following those plotted.

Note that LINE ends with a PLT or special SYMBOL call at the end of the line being plotted and AXIS ends with a NUMBER call labelling the last tick.

RFACT real output variable to return the current value of the plotting factor.

COMMON BLOCKS USED

CALCUR, CALCNT

EXIT

After calculating the coordinates (RX,RY) from the current pen position and retrieving FACT, control is returned to the calling routine.

EXAMPLE

The use of the routine WHERE is illustrated in the program example provided for the routine LINE.

AUTHOR L.E. Evans DATE January 1976.

TITLE

Calcomp routine to control pen position and plotting origin and to draw straight line segments (our local version of standard Calcomp routine PLOT).

ENTRY

CALL PLT(X,Y,IPEN)

X,Y — real input variables specifying the coordinates of a new pen position. A (real) value of 999. for either coordinate means that the corresponding coordinate of the current pen position is to be used.

IPEN — integer input control variable defined as follows:

If IPEN < 0, redefine the origin to be at the new pen position.

If |IPEN| = 2, draw a straight line segment from previous pen position to new pen position.

For 2<|IPEN|<999, move the pen to the new pen position but do not draw the connecting line (usually IPEN=±3 is used for this option).

If IPEN = 999, redefine the origin to be at the new pen position and flush the plotting buffers. This should be the last Calcomp routine called by a user's program.

ROUTINES CALLED

PLOTC from AELIB; PLFL, a utility routine, which is loaded with PLT.

LABELLED COMMON BLOCKS USED

CALCUR, CALCNT, CALBUF

EXIT

Normal Exit

After updating pen coordinates and performing other functions specified by IPEN, control is returned to the calling routine.

Error Exit

If either X or Y are illegal (i.e. indefinite or infinite), the error message

***ILLEGAL PEN COORDINATES - JOB TERMINATED

is issued with traceback and the job is terminated.

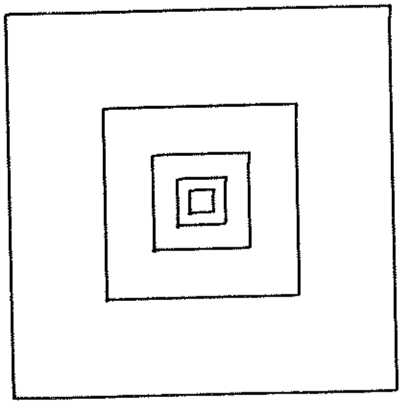
PROGRAM EXAMPLE

This program plots 5 concentric squares, illustrating the use of PLT and FACTOR.

```
      PROGRAM SQUARE(INPUT,OUTPUT,PLOT)
      REAL X(4),Y(4)
      DATA N/0/
      DATA X/0.0, 2.0, 2.0, 0.0/
      DATA Y/2.0, 2.0, 0.0, 0.0/
C
C     INITIALIZE PLOTTING SURFACE
C
      XL=2.0
      YL=2.0
      CALL PLOTCI(XL,YL)

      DELTA=.5
      FACT=1.0
C
C     PLOT SQUARE
C
 10   DO 50 I=1,4
         CALL PLT(X(I),Y(I),2)
 50   CONTINUE
      N=N+1
      IF(N.GE.5) GOTO 900
C
C     MOVE ORIGIN AND CHANGE PLOTTING FACTOR
C
      CALL PLT(DELTA,DELTA,-3)
      FACT=FACT/2
      CALL FACTOR(FACT)
      GOTO 10
C
C     RESET ORIGIN TO (0,0) AND FLUSH BUFFERS
 900  XO=0.0
      YO=0.0
      CALL PLT(XO,YO,999)
      STOP
      END
```

PLT



AUTHOR	L.E. Evans	DATE	January 1976
REVISED	L.E. Evans	DATE	April 1978

TITLE Calcomp routine to plot character text or special symbols.

ENTRY CALL SYMBOL(X,Y,H,ITEXT,A,N)

If N is Non-Negative (Plot Characters)

X,Y real input variables specifying the coordinates of the lower left-hand corner of the first character to be plotted. A (real) value of 999. for either coordinate means that the corresponding coordinate of the current pen position is to be used.

H real input variable specifying desired unscaled height in inches of the characters to be plotted. The actual size of the characters produced depends on whether the angle at which the characters are plotted is an even or odd multiple of 45°. If odd, the character size is $\sqrt{2}$ times the multiple of .08" closest to H*FACT if H*FACT lies between .05" and .84", and $\sqrt{2}$ x .24" if H*FACT lies outside this range. If even, the character size is the multiple of .08" closest to H*FACT if H*FACT lies between .05" and .84" and .24" if H*FACT lies outside this range. (FACT is the plotting factor described in the introduction to the Calcomp routines.)

ITEXT real or integer, input variable or the first word of an array containing the display code representation of characters to be plotted. If N > 0, this representation must be left justified. If N = 0, a single right justified character is assumed.

A real input variable specifying desired angle in degrees from the the x-axis, at which characters are to be plotted. The actual plotting angle is the next highest multiple of 45° between 0° and 360°. (Negative angles are converted to their positive equivalents before being rounded up.)

N — integer input variable specifying the number of characters to be plotted as follows:

N = 0 means that the right-most character is to be plotted from display code in ITEXT.

N > 0 means that the N left-most characters are to be plotted from the display code starting in ITEXT.

If N is Negative (Plot Special Symbols)

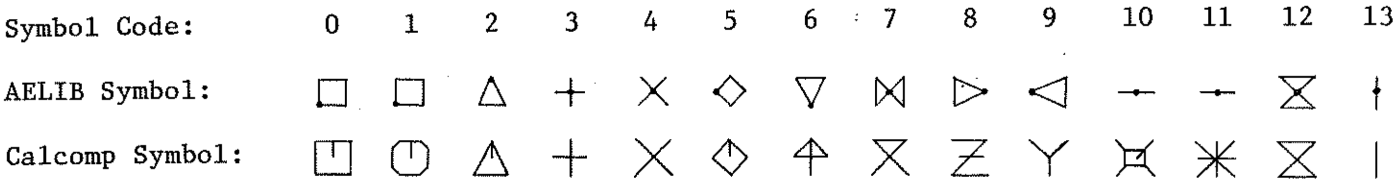
X,Y — real input variables specifying the coordinates of the starting point of the symbol to be plotted. A (real) value of 999. for either coordinate means that the corresponding coordinate of the current pen position is to be used.

H,A
ITEXT — real input variables and integer input variable to be used as follows:

If A is zero, ITEXT specifies a special symbol code from among those used by AELIB routine PLOT and H is unused.

If A is non-zero, ITEXT specifies special symbol code from among Calcomp's special symbols and H specifies the unscaled height of the symbol required. If the symbol requested is supported by the AELIB routine PLOT, then that symbol is plotted, in one of two sizes depending on the value of H*FACT. If H*FACT < .08", then symbol will be .04" in size and if H*FACT ≥ .08", then symbol will be .08" in size. If the symbol requested is not supported by PLOT, an AELIB equivalent symbol, as defined below, is plotted. (FACT is the plotting factor described in the introduction to the Calcomp routines.)

AELIB equivalent symbols for Calcomp symbol codes are:



N integer input variable used as follows: If N is less than or equal to -2, then a straight line segment is drawn from the previous pen position to the new pen position before the symbol is plotted. If N is equal to -1, no line is drawn.

ROUTINES CALLED PLOTC and PLT from AELIB; PLFL is a utility routine loaded with SYMBOL.

COMMON BLOCKS USED CALBUF, CALCUR, CALCNT

EXITS Normal Exit

After drawing character strings or special symbols and storing new pen position, control is returned to the calling routine.

Error Exit

If either X or Y are illegal (i.e. indefinite or infinite) the error message

***ILLEGAL PEN COORDINATES - JOB TERMINATED

is issued with traceback and the job is terminated.

EXAMPLE The use of the routine SYMBOL is illustrated in the program example provided for the routine LINE.

AUTHOR L.E. Evans DATE January 1976

TITLE

Calcomp routine to plot floating point numbers to desired accuracy in either F or I type FORTRAN formats.

ENTRY

CALL NUMBER(X,Y,H,FPN,A,N)

X,Y real input variables specifying the coordinates of the lower left-hand corner of the first character of the formatted output to be plotted. A (real) value of 999. for either coordinate means that the corresponding coordinate of the current pen position is to be used.

H real input variable specifying desired unscaled height, in inches of the formatted output to be plotted. The actual height of the characters produced is the same as that defined for character plotting from the routine SYMBOL (See SYMBOL writeup for details).

FPN real input variable specifying the floating point number to be converted and plotted.

A real input variable specifying the angle, in degrees from the x-axis, at which formatted output is to be plotted. The actual plotting angle is the same as that defined for character plotting from the routine SYMBOL (See SYMBOL writeup for details).

N integer input variable defining the accuracy and the type of format to be used for output of FPN.

If $N \geq 0$, then an F-type format is to be used with N decimal digits to the right of the decimal point.

If $N < 0$, then an I-type format is to be used to output the integer obtained by rounding FPN to the nearest integer and then truncating the $-(N+1)$ rightmost digits.

If $N < -9$, then $N = -9$ is assumed.

If $N > 9$, then $N = 9$ is assumed.

ROUTINES CALLED
PLOTC from AELIB; PLFL is a utility routine which is loaded with NUMBER.

COMMON BLOCKS USED
CALBUF, CALCUR, CALCNT

EXITS

Normal Exit

After plotting the floating point number to desired accuracy and storing new pen position, control is returned to the calling routine.

Error Exit

If either X or Y are illegal (i.e. indefinite or infinite), the error message

***ILLEGAL PEN COORDINATES - JOB TERMINATED

is issued with traceback and the job is terminated.

EXAMPLE
The use of the routine NUMBER is illustrated in the program example provided for the routine LINE.

AUTHOR L.E. Evans DATE January 1976.

TITLE

Calcomp routine to calculate from the data in an array, a starting value and a scaling factor, such that the data will fit and be well represented on the length of axis specified. This starting value and scaling factor are required by the Calcomp routines AXIS and LINE.

ENTRY

CALL SCALE(X,AX,N,I)

X(NRX) real input array, the data array for which scaling factors are to be calculated. $NRX \geq N*|I|+|I|+1$

AX real input variable specifying, in inches, the length of axis to which data is to be scaled.

N integer input variable specifying number of data entries to be examined in X.

I integer input variable used as follows:

$|I|$ is the increment at which data values are selected from X. The values selected will be X(1), X(|I|+1), X(2*|I|+1),etc.

If I is positive, then the starting value approximates the minimum of the data and the scaling factor is positive.

If I is negative, then the starting value approximates the maximum of the data and the scaling factor is negative.

EXIT

Error Exit

If SCALE is unable to calculate scaling factors that span the whole range of data supplied, the message

```
***SCALING UNABLE TO SPAN WHOLE RANGE
     MIN=       MAX=       AXIS LEN=
```

is printed with traceback and execution continues with the best values that SCALE can provide.

Normal Exit

The starting value is calculated and stored at $X(N*|I|+1)$ and the scaling factor is calculated and stored at $X(N*|I|+|I|+1)$. Control is returned to the calling routine.

EXAMPLE The use of the routine SCALE is illustrated in the program example provided for the routine LINE.

AUTHOR L.E. Evans DATE January 1976.

TITLE

Calcomp routine to draw any length axis at any angle, to draw and label ticks at one inch intervals and to plot an axis title.

ENTRY

CALL AXIS(X,Y,ITEXT,N,AX,A,FIRSTV,DELTAV)

X,Y — real input variables specifying the coordinates of the starting point of the axis to be drawn. A (real) value of 999. for either coordinate means that the corresponding coordinate of the current pen position is to be used.

ITEXT — real or integer input variable or the first word of an array containing the left justified display code representation of the desired axis title. This will be centered and plotted parallel to the axis in characters .16" high.

N — integer input variable used as follows:

|N| specifies the number of characters in the title and if N is positive, all labelling is done on the counter clockwise side of the axis; if N is negative, all labelling is done on the clockwise side of the axis.

AX — real input variable specifying the length of the axis in inches.

A — real input variable specifying the angle, in degrees from the x-axis, at which the axis is to be drawn. Note that labelling of the axis will be done parallel to the x-axis if the angle specified is not one of the eight angles supported by AELIB's PLOT.

FIRSTV — real input variable specifying the starting value for the first tick on the axis. This may be calculated by the user or by the routine SCALE.

DELTAV real input variable specifying the number of data units per inch of axis. This may be calculated by the user or by the routine SCALE.

ROUTINES CALLED PLT, NUMBER, and SYMBOL from AELIB.

COMMON BLOCKS USED CALCUR, CALCNT

EXIT After drawing and labelling the axis, control is returned to the calling routine.

EXAMPLE The use of the routine AXIS is illustrated in the program example provided for the routine LINE.

AUTHOR L.E. Evans DATE January 1976.

TITLE

Calcomp routine to draw a scaled line plot through a set of data points with the option of plotting special symbols at the data points.

ENTRY

CALL LINE(X,Y,N,I,LTYP,ISYMB)

X — real input array containing the x-coordinate data values followed by the two scaling parameters assumed to have been previously calculated for the data (see SCALE writeup).

Y — real input array containing the Y-coordinate data values followed by the two scaling parameters assumed to have been previously calculated for the data (see SCALE writeup).

N — integer input variable specifying the number of data points to be plotted from the X and Y arrays. Note that this does not include the two scaling parameters.

I — integer input variable specifying the increment to be used to retrieve data from the X and Y arrays, as described for the subroutine SCALE.

LTYP — integer input variable specifying the type of line plot to be drawn as follows:

(a) If LTYP is zero, then data points are joined by line segments, and no special symbols are used.

(b) Otherwise, special symbols are plotted at every |LTYP|th data point and if LTYP < 0, data points are not joined; if LTYP > 0 data points are joined by line segments.

ISYMB — integer input variable specifying special symbol to be plotted as follows:

If ISYMB is negative, |ISYMB| specifies a special symbol code from among those used by AELIB's routine PLOT.

If ISYMB is positive, |ISYMB| specifies one of Calcomp's special symbols as described for the subroutine SYMBOL. (Symbol height assumed is .08".)

ROUTINES CALLED

PLT and SYMBOL from AELIB.

EXIT

Normal Exit

After drawing the line plot, control is returned to the calling routine.

Error Exit

If either X or Y scaling parameters are illegal (i.e. infinite or indefinite), the message

***ILLEGAL SCALING PARAMETERS - JOB TERMINATED

is printed with traceback, and the job is terminated.

EXAMPLE

This program produces a line plot of test data with x coordinates increasing and y coordinates decreasing. Axes and a title are provided as well as a tag label on the curve. The plotted output (reduced to ½ size) is shown following the listing.

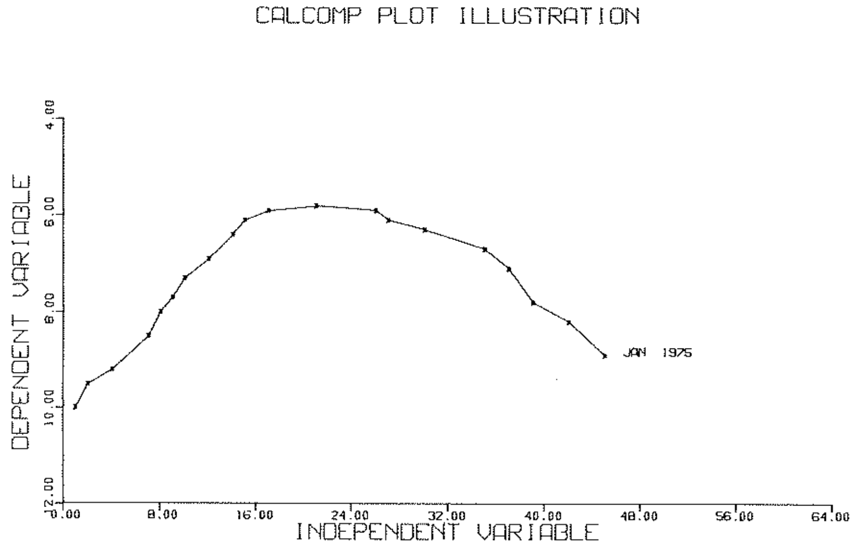
```
      PROGRAM CALC(INPUT,OUTPUT,TAPE1=INPUT,PLOT)
      REAL X(102),Y(102)
      INTEGER ITX(5),ITY(5)
C
      DATA X/1.0,2.0,4.0,7.0,8.0,9.0,10.0,12.0,14.0,15.0,17.0,21.0,
     ,26.0,27.0,30.0,35.0,37.0,39.0,42.0,45.0/
      DATA Y/10.0,9.5,9.2,8.5,8.0,7.7,7.3,6.9,6.4,6.1,5.9,5.8,5.9,6.1,
     ,6.3,6.7,7.1,7.8,8.2,8.9/
C
      I=20
C
C     INITIALIZE PLOTTING SURFACE TO 10 IN. BY 10 IN.
C
      XL=10.
      YL=10.
      CALL PLOTCI(XL,YL)
```

```
C
C     RELOCATE ORIGIN AT ( 1.0 , 1.0 )
C
      CALL PLT(1.0,1.0,-1)
C
C     PLOT TITLE
C
      ITX(1)=10HCALCOMP PL
      ITX(2)=10HOT ILLUSTR
      ITX(3)=10HATION
      XP=2.
      YP=5.
      H=.16
      A=0.
      N=25
      CALL SYMBOL(XP,YP,H,ITX,A,N)
C
C     SCALE X-COORDINATES TO 8 IN., X VALUES INCREASING
C
      XAX=8.
      INCX=1
      CALL SCALE(X,XAX,I,INCX)
C
C     SCALE Y-COORDINATES TO 4 IN., Y VALUES DECREASING
C
      YAX=4.
      INCY=-1
      CALL SCALE(Y,YAX,I,INCY)
C
C     PLOT AXES
C
      ITX(1)=10HINDEPENDEN
      ITX(2)=10HT VARIABLE
      NX=-20
      AX=0.
      ITY(1)=10HDEPENDENT
      ITY(2)=10HVARIABLE
      NY=18
      AY=90.
      XP=0.
      YP=0.

      CALL AXIS(XP,YP,ITX,NX,XAX,AX,X(I+1),X(I+2))
      CALL AXIS(XP,YP,ITY,NY,YAX,AY,Y(I+1),Y(I+2))
C
C     PLOT LINE WITH SMALL X AT EACH POINT
C
```

```
      INC=1
      LTYP=1
      ISYMB=-7
      CALL LINE(X,Y,I,INC,LTYP,ISYMB)
C
C     TAG LINE WITH LABEL     JAN 1975
C     USING BOTH SYMBOL AND NUMBER CALLS FOR ILLUSTRATION
C     SYMBOL ALONE WOULD HAVE SUFFICED FOR THE TEXT
C     WHERE IS USED TO LOCATE THE STARTING POSITION OF
C     THE LAST SPECIAL SYMBOL
C
      CALL WHERE(XP,YP,FACT)
      XP=XP+.2
      H=.08
      A=0.
      N=4
      ITX=4HJAN
      CALL SYMBOL(XP,YP,H,ITX,A,N)
      XP=999.
      YP=999.
      N=-1
      FPN=1975
      CALL NUMBER(XP,YP,H,FPN,A,N)

      STOP
      END
```



AUTHOR L.E. Evans DATE January 1976

2- 2 INPUT/OUTPUT ROUTINES (EXCLUDING PLOTTING)

00-09	FREE FORMAT CARD INPUT		
	2-2-00	FREEIN	FREE FORMAT INPUT ROUTINE
10-29	MAGNETIC TAPE DATA STORAGE		
	2-2-10	DATPAK	DATA MANAGEMENT PACKAGE
30-39	PAPER TAPE INPUT AND OUTPUT		
	2-2-30	PTREAD	PAPER TAPE READING ROUTINE
	2-2-31	PTPUN	PAPER TAPE PUNCHING ROUTINE

TITLE Free Format Input Routine

ENTRY CALL FREEIN(INUNIT,ARRAY,NWD,MODE,NUSED)

INUNIT integer input variable, the unit number from which (card) input is to be read. INUNIT must refer to coded file containing records of up to 140 characters.

ARRAY real output array which stores the data being read.

NWD integer input variable specifying the maximum number of words to be read into ARRAY.

MODE integer input variable, 0 if only one card image is to be read; 1 if several card images are to be read.

NUSED integer output variable, the actual number of words read into ARRAY.

ADDITIONAL ENTRY INFORMATION

i) Types of Input

(a) INTEGER NUMBERS: e.g. 5, -991, +22, $-777. Note: A $ sign before a number indicates an octal number. Integer input data is converted to floating point format.

(b) FLOATING POINT NUMBERS: e.g. -11.975, $-567E-07, +12.1.

(c) ALPHANUMERIC STRINGS: These must start with a letter. Characters are stored left justified with blank fill. After a separator is encountered, the previous string is placed in the array with the next character read starting a new word; e.g. AN EXAMPLE would be assembled into two 6600 words, AN in the first and EXAMPLE in the second, both left justified blank filled.

(d) COMMENTS: A whole card image can be stored in the next eight words of ARRAY if an * is punched in card column 1. The * will be replaced by a blank. A comment enclosed in parentheses is ignored and will not be stored in ARRAY; e.g. (THIS COMMENT (1) WILL BE IGNORED).

Note: Double precision and complex numbers are not allowed.

ii) Termination of Field

Fields are terminated by any of the following:

Blank ( )
Equals sign (=)
Comma (,)
Slash (/)
or at the end of a card.

iii) Termination of Input

Input is terminated under the following conditions:

(a) On encountering a slash (/) in the input string.
(b) When NWD words have been read.
(c) When an end of file is read on the input unit. If NUSED = -1, no data was transferred, only an EOF was read.
(d) When MODE = 0, only one card image is read.

EXIT

Data is stored sequentially in ARRAY with NUSED containing the number of input data words stored. Diagnostics may be reported during processing as follows:

(1) The maximum size of ARRAY is reached. A message stating the card number and the column number of the last item assembled is printed.

(2) A number is too large to be stored in ARRAY. The position of the number on the card input is printed in an error message, and the number is ignored.

(3) If an end of file is encountered on one call of FREEIN, the next call will print a message reporting this and then carry on processing.

SPEED Typical times:

.163 sec for 705 integer numbers)
.264 sec for 500 floating point numbers) 6600, November 1972
.047 sec for 100 words of comments)

AUTHOR C.D. Price DATE November 1972

TITLE DATPAK - Data Management Package (Version 2)

INTRODUCTION The subroutine DATPAK allows users a means of storing data and associated comments on a sequential tape file, in an indexed format. DATPAK allows for easy access and modification using the generated file and data block numbers issued during file manipulations.

Any program using DATPAK must have the file TAPE6, the default DATPAK output file, declared on its program card, or else change this unit number with a call to DPUNIT.

DATPAK makes direct calls to Record Manager. No references to DATPAK data files are allowed on the PROGRAM card. DATPAK creates SI (System Internal) blocked binary tapes with WI-type records.

ENTRIES There are 13 possible entries to DATPAK. These are described following the list of formal parameters used by more than one call.

FORMAL PARAMETERS

IOT integer input variable, the logical unit number of DATPAK tape. If the logical file name used on the control card request for the DATPAK tape is:

i) TAPE3, then IOT=5LTAPE3 or IOT=3
ii) DATA, then IOT=4LDATA.

LCOMS integer input or output variable, the number of words of comments to be retrieved or added, set to 0 if unknown for DPREAD.

LAB integer input or output array containing comments.

LDAT integer input or output variable, the number of words of data to be added or retrieved, set to 0 if unknown for DPREAD.

DAT real or integer input or output array containing data.

NF integer file number, returned from DPADD, file to be read for DPREAD.

NDB integer data block number, returned from DPADD, data block to be read for DPREAD.

MODE integer input variable:

for DPREAD if 0, NF, NDB returned.
if 1, next data block returned from current position.

for DPADD if 0, message issued to output unit after data block added.
if 1, expanded output format;
- messages issued before and after data block is added.
- comments and tape footage also listed.

IVAL integer output variable; in DPREAD it is a flag indicating the reason for termination of read (see DPREAD).

IQUICK integer input variable;
if 0, forces a DATPAK end of information record to be written after every DPADD.
if 1, forces quick add mode which suspends writing an EOI after each DPADD (less safe in case of program fault or system crash).

CALLING SEQUENCES

(1) Initiate DATPAK Tape

```
CALL DPLABEL(IOT,LCOMS,LAB)
```

Every DATPAK tape must be initialized by a call to DPLABEL. LCOMS words from the user's array LAB are written onto logical unit IOT.

(2) Addition of Data

CALL DPADD(IOT,LCOMS,LAB,LDAT,DAT,NF,NDB,MODE,IQUICK)

LCOMS words from the user's array LAB, and LDAT words from the user's array DAT, are written at the current end of information onto logical unit IOT. DATPAK will return the file and data block numbers in NF and NDB. If MODE is set to 1, expanded output will be issued; otherwise with MODE set to 0, the user is informed after each successful add. If IQUICK is set to 1, the quick mode for addition is chosen, otherwise with IQUICK set to 0, a DATPAK end of information record is written after every DPADD. This will always ensure that a DATPAK tape is properly terminated.

(3) File Number Increment

CALL DPFILE(IOT)

This will cause a file mark to be written at the current end of information onto logical unit IOT, and increment the file counter by one.

(4) Read a Data Block

CALL DPREAD(IOT,LCOMS,LAB,LDAT,DAT,NF,NDB,MODE,IVAL)

If MODE is set to 0, then file NF data block NDB will be read from the DATPAK tape on logical unit IOT. DATPAK will return the comments retrieved into the user array LAB and the data into user array DAT, with LCOMS and LDAT containing the number of words retrieved in each.

If the number of words of comments (LCOMS) and/or the number of words of data (LDAT) are set to non-zero values before the call, then only the first LCOMS or LDAT words are retrieved from the specified file and data block. If LCOMS and LDAT are set to zero, then all words of comments and data are retrieved.

If MODE is set to 1, the next data block from current position is retrieved.

IVAL will contain an integer number indicating the reason for termination of the read.

0 - read was successful
1 - parity error recoverable
2 - parity error unrecoverable
3 - file mark read (MODE=1)
4 - data block not found

(5) Read DATPAK Label Record

```
CALL DPREADL(IOT,LCOMS,LAB)
```

LCOMS words are stored in the user array LAB, read from the DATPAK tape on logical unit IOT.

(6) List Contents of DATPAK Tape

```
CALL DPLIST(IOT)
```

The entire contents of the DATPAK tape on logical unit IOT is listed on the DATPAK output file. The listing will include the DATPAK label, each comment block and its file and data block index. At completion the total tape footage count is displayed.

(7) Change Default Output Unit

```
CALL DPUNIT(MUNIT)
```

The DATPAK output file is changed from TAPE6 to unit MUNIT by this call.

(8) Change Tape Recording Density

```
CALL DPDNSTY(IOT,IDENS)
```

This call will change the default recording density from 800 bits per inch to IDENS on unit IOT. This value is used only to compute the tape footage used at DATPAK end of information. Only values of 200, 556 and 1600 bits per inch are acceptable for IDENS.

(9) Write End of Information

```
CALL DPWEOI(IOT)
```

The user must call DPWEOI after completing the final DPADD using the quick add mode feature only. This ensures that the DATPAK tape on logical unit IOT has been properly terminated.

(10) Reset End of Information

CALL DPRESET(IOT,NF,NDB)

After an abnormal termination of a DATPAK run, a call to DPRESET must be issued after the last successfully written data block, before any other operation is performed on the DATPAK tape. DPRESET may also be used to reposition the end-of-information record, in order to over-write data no longer needed. End of information is written after file NF data block NDB, on logical unit IOT.

(11) Error Recovery

CALL DPEROPT(IOT,IEROPT,ITRIV)

This call allows the user to control error recovery procedures to be used when tape reading errors are encountered. IEROPT specifies the recovery option and ITRIV specifies the number of times error recovery shall take place before the run is terminated. Refer to User's Guide for the values and interpretation of IEROPT.

(12) Buffer Area

CALL DPBUFST(IOT,IBUFSZ,FWA)

This call defines a buffer area for the DATPAK data file (IOT) of size IBUFSZ starting at FWA. Buffer space is normally assigned to BLANK COMMON by DATPAK and this call is necessary only to change this default.

(13) Access Termination

CALL DPCLOSE(IOT,IOPTION)

This call allows the user to terminate access to the DATPAK tape IOT. IOPTION indicates whether the DATPAK tape is to be unloaded, rewound, or left at its current position: 1LU, 1LR, 1LN respectively.

EXIT

A normal exit returns control to the calling program. Error diagnostics are issued when abnormal conditions exist and certain flags are set, particularly in DPREAD. All messages issued by DATPAK are preceded by an asterisk(*). The first call to any DATPAK entry point first checks the validity of the DATPAK tape. The subroutine will issue a STOP if your tape is not a DATPAK tape.

AUTHOR	C.D. Price	DATE	August 1974
	Revised by H. Keech		May 1978

TITLE — Paper Tape Read Subroutine.

INTRODUCTION — The subroutine PTREAD has been provided to process paper tapes read by the paper tape reader attached to the 3300.

This writeup provides information on how to call PTREAD and should be used in conjunction with the document MCTD-6, "A Guide to Paper Tape Input on the CDC 6600/CYBER 170", that provides introductory material on paper tape reading, definitions of terms used herein, description of paper tape formats and coding, and a sample FTN program using PTREAD.

Any program which uses PTREAD must have the following declaration of the PROGRAM card:

```
       PROGRAM TEST(...,...,ptfile)
```

where ptfile is PTREAD unless the user specifies some other file name via a MODE6 call.

If a paper tape is submitted with the job deck, then the 170/6600 job card must have PT as one of the parameters.

All arguments except those marked by an asterisk must be of type INTEGER. The others can be INTEGER or REAL within the limitations mentioned in the descriptions.

ENTRIES

```
CALL PTREAD(0)

CALL PTREAD(1,ISC,IEC,ITYPE,IPAR,NBITS,NBLANKS,KODARR)

               *
CALL PTREAD(2,FLIST)

                  *
CALL PTREAD(3,NR,ARRAY,NUM1,NUM2,IERR)

CALL PTREAD(4,N)

CALL PTREAD(5,IARR,NWD,ISTAT)

CALL PTREAD(6,LFN)
```

Each entry with its formal parameters is described below:

1. Dump Input Buffer

   CALL PTREAD(MODE)

   MODE=0

   Turn on switch to dump each buffer as it is read. The next call turns the switch off.

   The data is dumped in words of three octal digits, 30 to a line.

2. Initial Set Up

   CALL PTREAD(MODE,ISC,IEC,ITYPE,IPAR,NBITS,NBLANKS,KODARR)

   MODE=1

   ISC - octal code indicating the start of a record;
   IEC - octal code indicating the end of a record.

   The absence of either of these codes is indicated by a zero value. In such instances, the start and end of a record are determined as follows:

	Start Code on First Read	Start Code on Intermediate Read	End Code on Intermediate Read	End Code on on Final Read
ISC=0 IEC≠0	PHYSICAL START OF TAPE	IEC	IEC	IEC
ISC≠0 IEC=0	ISC	ISC	NBLANKS FRAMES	PHYSICAL END OF TAPE
ISC=IEC =0	PHYSICAL START OF TAPE	NBLANKS FRAMES	NBLANKS FRAMES	PHYSICAL END OF TAPE

ITYPE	0 for coded tape; 1 for binary tape.

IPAR 0 if no parity checking is required;
N if parity checking is required, where N specifies the channel containing the parity bit, positive for even parity, negative for odd parity.

NBITS For a coded tape, this specifies the number of channels of information to be used when decoding the information.

For a binary tape, this specifies the number of channels containing binary data. The lower consecutive NBITS are used.

If zero, NBITS is set to 6 (default).

NBLANKS The minimum number of consecutive blank frames of tape to be considered as start and end codes of a record. Runs of fewer than NBLANKS consecutive frames are ignored throughout the tape.

If zero, NBLANKS is set to 10 (25.4 mm (1 in.) of tape).

KODARR The decoding matrix used to convert paper tape codes (up to 8 bits) to 6 bit 6600 display code.

For an ASCII coded tape containing only numbers, the easiest way to set up KODARR is:

```
DIMENSION KODARR(8)
DATA KODARR/6*0,3334353637404142B,4344000000000000B/
```

The details for setting up this matrix are given in Appendix I of MCTD-6.

Note: Further calls of Mode 1 may be made to redefine any of the parameters. The tape is not repositioned.

3. Formatting Specification

CALL PTREAD(MODE,FL$\overset{*}{\text{I}}$ST)

MODE=2

FLIST is an array containing a formatting specification. There are two types of formatting specifications allowed; one for coded tapes and one for binary tapes:

(a) Coded Tapes

The specification is similar to the FORMAT statements used in FORTRAN; e.g.

```
(3X,I6,1X,A10,4(6O7,5X))
```

The field descriptions allowed are X, A, R, O, I, F, E, C (see below).

nX means that n DATA codes are to be skipped (NON-DATA codes are always skipped). Leading blanks are permitted in fields read under O, I, F or E field descriptors.

O, I, E and F field descriptors are consistent with other FORTRAN conventions. For E and F field descriptors, the location of the decimal point or exponent punched on the tape overrides the position specified for these in the field descriptor.

Cm (check character) means that the frame denoted by m (octal) must appear on the tape in the position it occupies in the format statement. Failure to find it indicates an error and the read stops at this point. The characters following (up to 100 in number) may be obtained by CALL PTREAD(4,2). For maximum value to be obtained from this feature, m must not be one of the DATA codes; i.e. it may be considered a special kind of NON-DATA code.

A formatting specification is most easily generated using a DATA statement as follows:

```
DIMENSION FLIST(2)
DATA FLIST/19H(nX,nAm,nIm,nFm,nCm)/
```

(b) Binary Tapes

The source of paper tapes containing binary information is often computer-based data-gathering systems, and the binary data usually consist of integer numbers dumped directly as they were stored in core memory. The binary data usually occupy the lower six bits in successive frames of the eight-channel paper tape. Channels seven and eight are generally used for flag bits which, for example, may indicate the start of a data word or act as check characters. The order in which the successive frames of binary data must be assembled has to be specified since the order in which the data are dumped to tape is not necessarily the order to be used when assembling the word.

Thus, the main body of the formatting specification consists of pairs of words, the first of which specifies a mask which is used to identify a particular frame of tape. E.g. If a punch in channels 7 and 8 is used to flag the first frame of binary data, then the mask would be specified as 300B.

The second word of the pair specifies the position of the NBITS of data in the final computer word. The least significant position will be zero.

There will be one pair of words for each frame of data on tape. Blank frames are ignored unless there are more than NBLANKS. NBLANKS should be set greater than the number of blanks likely to be encountered within a word.

Check characters are designated by the uncoded values with a negative sign. A blank cannot be used as a check character. The octal code 777B signifies the end of the format.

A sample formatting specification for a binary tape might be

```
DIMENSION FLIST(10)
DATA FLIST/300B,1,200B,0,200B,3,200B,2,-1B,777B/
```

Note: Mode 2 may be called repeatedly with different formats as required.

4. Data Read

CALL PTREAD(MODE,NR,ARRAY*,NUM1,NUM2,IERR)

MODE=3

NR =0, next start code signifies the start of the record.

=1, skip to the beginning of a new physical tape. Except for this option, there will be no recognition of the end of one tape and the start of a new one except for the blank frames intervening.

=-1, next read continues immediately after the previous one without looking for a start code. This is used mainly with tapes for which ISC and IEC cannot be defined.

=-m, backspace the tape 10(m-1) frames and read forward from that point without searching for a start code (where $2 \leq m \leq 16$).

ARRAY array to store the data. It must be REAL if the format described in FLIST is E or F, otherwise it must be INTEGER.

NUM1 number of data words to be read. If NUM1=0, reading continues until an IEC is encountered. NUM1 is usually set to the size of ARRAY.

NUM2 number of words actually read. This variable is returned by the subroutine.

IERR error indicator word. A bit is set in this word for any of the following errors:

Decimal Value	Octal Value	Bit	Coded Tape Error	Binary Tape Error
1	1	0	Too few digits in data word or invalid characters encountered in data word.	Incorrect mask.
2	2	1	Check characters do not match.	Incorrect check character.
4	4	2	Parity error.	Unassigned.
8	10	3	End of one tape (not the end of all tapes) has been reached.	Same as for coded tape.
16	20	4	Physical end of all tapes has been reached. Note that bit 4 entails also bit 3.	Same as for coded tape.
32	40	5	Attempt to call PTREAD after physical end of tapes has been reached.	Same as for coded tape.

The user should check IERR after every read. Note that several error bits may be set simultaneously; the corresponding values are sums of the individual values.

If bit 0 or 1 is set, an indefinite quantity is stored in ARRAY.

The first ten words for coded tapes or the first 20 words for binary tapes containing errors are saved in an array which may be dumped with a PTREAD(4,1) call.

5. Diagnostic Dump

CALL PTREAD(MODE,N)

MODE=4

N Specifies the type of dump.

N=1 Coded Tape

Dumps words in error. The number of characters is defined by the format statement (e.g. I6) or by the occurrence of a NON-DATA code or a run of blank frames, represented by #. Thus the following words are erroneous in an I6 format:

12E456 789#

the first because it contains an invalid character; the second because there are too few digits. The corresponding words in the output array are set to indefinite. If parity is being checked and an error is encountered, the parity bit is reversed, the frame is decoded and the result put in the output array. This assumes that the parity bit is in error (not necessarily the case). The error word contains the same as the output word with arrow(s) marking the wrong parity character(s).

$$12\overset{\downarrow}{3}45\overset{\downarrow}{6}$$

Up to ten error words are accepted.

Binary Tape

Each error sets the output word to indefinite. The error words (maximum 20) are made up as follows:

Octal digits 6-19 contain the output accumulated before the error was encountered.

Digit 4 contains the frame number which is in error.

Digits 0-2 contain the octal representation of the erroneous frame.

Digits 3 and 5 are zero.

N=2 For Use With Coded Tape Only

Dumps the decoded tape for 100 characters on each side of the present position. The last character examined by the program appears at the extreme right of the top line. NON-DATA characters are represented by #. The dump does not cross over the division between physical tapes.

6. Raw Data Read

CALL PTREAD(MODE,IARR,NWD,ISTAT)

MODE=5

This mode is incompatible with any other mode.

IARR integer array (length $\geq$ 64) into which the raw data is transferred. Five 12-bit bytes are packed in each 6600 word.

NWD number of words of information stored in IARR. It is returned by the subroutine. NWD=64 except for the last buffer of each tape when it may be less.

ISTAT -1 normal terminal;
0 end of a physical tape;
+1 end of all tapes.

The data are packed five frames to a 6600 word, each frame occupying the lower eight bits of each 12-bit byte. Blank frames are compressed up to 377_8 (255_{10}) in number, with bit 8 set. This is equivalent to adding 400_8 to the number of blank frames.

The number of blank frames preceding the first data frame varies since the operators may not load the tape at the same place each time, so for the sake of uniformity, the first byte of the first tape is always 710_8, corresponding to 310_8 (=200_{10}) blank frames. Blank frames are ignored on each new tape after the first. The number of blank frames at the end of each physical tape is left as read.

Each physical tape consitutes a system-logical-record. An end-of-file follows the last record. ISTAT is returned by the subroutine and should be checked by the user after every Mode 5 call.

Only Modes 0 or 6 may be used with Mode 5.

A detailed description of the structure of a paper tape input file is given in Appendix II of MCTD-6.

7. Change File Name 'PTREAD' to Name Supplied by User

CALL PTREAD(MODE,LFN)

MODE=6

LFN = integer variable containing the user-supplied file name, left justified with zero fill; e.g.

DATA LFN/5LTAPE1/

EXIT A normal exit returns control to the calling routine. Errors in the data punched on the tape or other non-fatal errors are flagged as described above and the job continues. Fatal errors, however, cause a diagnostic message to be issued to the output file and the job to be terminated. Diagnostics issued by PTREAD are of the form:

(Error Message)
AELIB ERROR, REPORTED AS TRACEB ERROR NUMBER 52, FORCES JOB TERMINATION.
ERROR NUMBER 52 DETECTED BY TRACEB AT LINE xxx CALLED FROM COMP AT LINE yyy

The possible error messages, and the PTREAD mode(s) from which they may arise are as follows:

Error Messages	PTREAD MODE
EMPTY PAPER TAPE FILE	3,5
ILLEGAL NUMBER OF PARAMETERS	0,1,2,3,4,5
INVALID CHARACTER(S) IN FORMAT STATEMENT	3
MODE NOT 0 TO 6	>6 or <0
CALL OF MODE3 WITHOUT PREVIOUS CALL OF MODE 1 OR 2	3
FIFTH ATTEMPT TO CALL PTREAD AFTER ENCOUNTERING EOF	3,5

AUTHOR	G.N. Williams J.F. Steljes	DATE	March 1973
REVISED	G.N. Williams J.F. Steljes	DATE	November 1975
	J.F. Steljes	DATE	August 1982

TITLE Punch Paper Tape from a FORTRAN Program.

ENTRY

```
CALL{PTPUN(IUN,IDATA,NWD,MODE,FORMAT,ICONV)
    {PTPUN(IUN,IDATA,NWD,MODE)
```

IUN integer input variable, the logical unit number of the file that is to be used as a punch paper tape file.

IDATA integer input array containing the data that is to be punched onto the paper tape.

NWD integer input variable, the number of words to be transferred from IDATA to the paper tape file. (Note: This parameter has a slightly different meaning when the binary punching option is used.)

MODE integer input variable containing a character string specifying punching mode: 2RPS if six-channel mode is required and 2RPT if eight-channel mode is required. The punching mode is determined on the first call to PTPUN and cannot be modified in subsequent calls.

Note: If parameter list is truncated after the MODE parameter, the binary punching option is used. In this case, the routine assumes that the words in IDATA have already been packed ready for punching. In eight-channel mode NWD 12-bit bytes are passed from the array IDATA to the punch file. With six-channel mode, NWD 6-bit characters are passed from the array IDATA to the punch file.

FORMAT integer or real input array used to pass the format to be used in punching the paper tape. The format decoder in the routine can handle A, I, L, O, R and X formats. Repeated format specification is not allowed; therefore, (2(3A1,2X)) must be passed as (3A1,2X,3A1,2X).

ICONV integer array of conversion codes from 6600 display code into the punching codes. This array is indexed by the 6600 display code; thus ICONV(1) should equal the code for A. If missing from the parameter list, no conversion from 6600 display code is performed.

Apart from supplying the correct calling parameters, the user need do nothing else. The routine automatically opens, fills and closes the punch file.

EXIT

A normal exit returns control to the calling program. However, should an error be encountered, a diagnostic message is entered in the job dayfile and the job is terminated. All the following errors cause the program to terminate following the issuing of the appropriate message in the job dayfile.

(a) PTPUN - INCORRECT CALL - error in the formal parameters.

(b) PTPUN - BUFFER NOT FOUND - file named in IUN is not found.

(c) PTPUN - BUFFER TOO SMALL - buffer reserved for the punch file is less than 1042B in length.

(d) PTPUN - ILLEGAL FORMAT PARAMETER TYPE - illegal format letter is encountered.

(e) PTPUN - ILLEGAL FORMAT DEFINITION - described FORMAT contains illegal brackets.

SPEED

Depends on the amount of data in each call and the complexity of the format.

AUTHOR D.B. Goulding DATE December 1972

2- 3 TABULAR DATA MANIPULATION

00-09	LOCATING ENTRIES IN ORDERED TABLES		
	2-3-01	LOCRL	LOCINT TABLE LOOK-UP FUNCTION
10-19	SORTING NUMERICAL ARRAYS IN ORDER		
	2-3-10	SORTAG	DESORT SORTING DATA WITH TAG
	2-3-11	ORDERF	ORDERB SORTING DATA WITH TAG ARRAY + ERROR FLAG
30-39	INITIALIZING MEMORY TO DESIRED BIT PATTERNS		
	2-3-30	PRESET	INITIALIZE TO USER-DEFINED PATTERN
	2-3-31	RESET	RESET CORE TO LOADER PATTERN

2-3 TABULAR DATA MANIPULATION

Searching and Sorting

The AELIB function LOCRL and subroutines SORTAG and ORDERF (with additional entries LOCINT, DESORT and ORDERB respectively) provide basic searching and sorting capabilities. LOCRL does table look-up by comparing the value to be located with the sorted table entries in the order first to last or last to first. While this algorithm is adequate for doing occasional searches, applications with extensive searching requirements could warrant a special purpose searching routine. For a discussion of searching techniques, see a general reference such as [1].

SORTAG is a general purpose sorting routine using a tag array to record the reordering of items. This tag array can be very useful, for example, if several arrays are to be reordered in parallel. The algorithm used in SORTAG is a version of QUICKSORT[2]. The two IMSL routines VSORT and VSORTA use another version of this algorithm. ORDERF/ORDERB is a copy of SORTAG/DESORT with an extra parameter to return an error flag value to the calling routine. A study of sorting techniques and testing of these routines has recently been carried out[3]. While QUICKSORT is a good general purpose algorithm, there may be applications in which a special purpose routine is warranted. For both searching and sorting, Mathematics and Computation Branch will, on request, help you to select the best algorithm for your application.

Core Setting and Resetting

The subroutine PRESET was originally written to initialize large arrays at the beginning of a program. However, it is currently used far too often and in places where a simple DO loop would be more efficient.

```
    ·CALL PRESET(S,I,P)
```

with S and P either both integer or both real, is logically equivalent to

```
     DO 10 J=1, I

        S(J)=P
  10 CONTINUE
```

To define variables and short arrays, the overhead of entry and exit from PRESET can be significant. Also, if PRESET is called many times in a job, even a small relative overhead can be costly. So before you use PRESET, consider seriously using a DO loop instead.

The subroutine RESET is for use with overlay programs. When an absolute file of an overlay program is created, default NOS/BE core presetting will first preset all overlays to negative infinites. The only area of core that is not guaranteed to be preset is any part of blank common which is obtained dynamically by calling CHNGSCM. RESET may be called after CHNGSCM to preset this section of core.

The pattern used by RESET is the old SCOPE default preset pattern of indefinites,

i.e. 60000 00000 04004 00000 + IJKLMN

where IJKLMN is the octal address of the word being set.

Be careful to use the right limits for RESET. It can overwrite code if limits are set improperly.

L.E. Evans
July 1979

REFERENCES

[1] D.E. Knuth, The Art of Computer Programming, Volume 3 Sorting and Searching, Addison-Wesley, Reading, Mass., 1973

[2] R.C. Singleton, An Efficient Algorithm for Sorting with Minimal Storage, Comm. ACM, 12(3),1969, pp. 185-187

[3] P.A. Christie and L.E. Evans, A Review of Computer Sorting Methods, AECL-6360, in preparation.

TITLE Table Lookup Function

ENTRY There are two entries to this routine

I = LOCRL(Y,X,N,IOPT,IER,TOL)

for table look-up in a real array, and

I = LOCINT(IY,IX,N,IOPT,IER)

for table look-up in an integer array,

where

Y is a real input variable specifying the value to be located in the table.

X is a real input array containing the table of values to which value of Y is to be compared.

IY is an integer input variable specifying the value to be located in the table.

IX is an integer input array containing the table of values to which the value of IY is to be compared.

N integer input variable specifying both the order and number of elements of X/IX to be searched. If N is positive, the N elements X(1) ... X(N)/IX(1) ... IX(N) are searched in that order. If N is negative, the -N elements X(N) ... X(1)/IX(N) ... IX(1) are searched in that order. N equal to zero is considered an illegal argument.

IOPT integer input variable specifying choice of option as follows:

1 locate first element in X/IX equal to Y/IY. For entry LOCRL, a non-zero value of TOL means that a table value of Y±TOL will be accepted as being equal to Y. Note that if TOL=0, the first element in X that is equal to Y will be selected.

2 locate first element in X/IX less than or equal to Y/IY.

3 locate first element in X/IX greater than or equal to Y/IY.

4 locate first element in X/IX closest to Y/IY.

IER integer output variable used to pass an error flag value back to the calling routine. IER has value zero if no errors occurred and may have a value of 1, 2 or 3 under error conditions specified in Error Exits below.

TOL real input variable specifying a tolerance for Y for entry LOCRL when IOPT=1. A tolerance value of zero is used if TOL is indefinite or infinite.

COMMON BLOCKS USED AELERCM

EXITS

Normal Exit

If no errors occur, the function value is set to the position in the array of the variable first satisfying the condition specified by IOPT, IER is set to zero, and control is returned to the calling routine.

Error Exits

(1) If no element is found in X/IX for IOPT=1, 2 or 3, IER is set to 1, the function value is set to zero and control is returned to the calling routine.

(2) If more than one element is found in X/IX closest to Y/IY for IOPT=4, IER is set to 2, the function value is set to the index of the first element found and control is returned to the calling routine.

(3) If any illegal arguments are encounted, that is if Y/IY or any entries examined of X/IX are infinite or indefinite, or N is zero or IOPT is not 1, 2, 3 or 4, then the message

***ILLEGAL ARGUMENT - FUNCTION VALUE SET TO ZERO

is printed, IER is set to 3, the function value is set to zero and control is returned to the calling routine.

TIMING

The maximum time required to search an array of 100 elements for IOPT = 1, 2 or 3 is approximately 2 ms and for IOPT = 4 is 3 ms. (6600, February 1976)

EXAMPLES

(1)
```
Y=.5
N=10
IOPT=2
J=LOCRL(Y,X,N,IOPT,IER,TOL)
```
searches 10 elements of the real array X, starting at X(1), for a value X less than or equal to .5.

(2)
```
TOL=.01
Y=6.
N=-20
IOPT=1
J=LOCRL(Y,X,N,IOPT,IER,TOL)
```
searches 20 elements of X, in the order X(20) to X(1), for a value of X equal to 6. ± .01.

(3)
```
IY=30
N=60
IOPT=4
J=LOCINT(IY,IX,N,IOPT,IER)
```
searches 60 elements of the integer array IX, starting at IX(1), for the element closest in value to 30.

AUTHOR D.G. Stewart Date February 1976

TITLE Sort Data with Tag Array

ENTRIES There are two entries to this routine:

CALL SORTAG(A,I,J,TAG)

sorts in increasing order and

CALL DESORT(A,I,J,TAG)

sorts in decreasing order.

A real or integer array containing elements to be sorted. On return from SORTAG(DESORT) the sorted array replaces the original contents.

I,J integer input variables specifying the sub-array of A, i.e. A(I), A(I+1), ..., A(J), to be sorted.

TAG real array whose elements are permuted along with the corresponding elements of A.

EXIT A call to SORTAG (or DESORT) returns the elements of A in order of increasing (or decreasing) size. The elements of TAG are reordered along with A.

If I≥J, control returns to the calling routine.

SPEED The speed of SORTAG depends on the number of elements to be sorted. The times required to sort random arrays are summarized below:

Length of Array	Time to Sort	(Cyber 170, Oct. 1977)
32	.65 ms	
512	18 ms	
8192	390 ms	

REFERENCE R.C. Singleton, An Efficient Algorithm for Sorting with Minimal Storage, Comm. ACM, 12(3),1969, pp. 185-187

WRITTEN BY C.J. Johnston DATE January 1970

DOCUMENTATION REVISED L.E. Evans DATE April 1978

TITLE Sort Data with Tag Array.

INTRODUCTION This routine sorts a data array and permutes a tag array in the same way. The ordering is done by integer subtraction, so if floating point numbers are to be sorted, they must be in normalized form.

ENTRIES There are two entries to this routine:

CALL ORDERF(A,I,J,TAG,IER)

sorts in increasing order, and

CALL ORDERB(A,I,J,TAG,IER)

sorts in decreasing order.

A real or integer array containing elements to be sorted. On return from ORDERF(ORDERB) the sorted array replaces the original contents.

I,J integer, input variables specifying the sub-array of A, i.e. A(I), A(I+1), ..., A(J), to be sorted.

TAG real or integer array whose elements are permuted along with the corresponding elements of A.

IER integer output variable used to pass an error flag value back to the calling routine. IER has a value of zero if no errors occurred and may have a value of 1 under the error condition specified in Error Exit below.

ROUTINES CALLED TRACEB and ALERPR from AELIB.

EXITS Normal Exit

If no errors occur, the arrays are permuted, IER is set to zero, and control is returned to the calling routine.

Error Exit

If the array indices I and J are not both greater than zero, or if I>J, then the message

***ARRAY INDICES I = __, J = __ ARE NEGATIVE, ZERO, OR I.GT.J.

is printed along with a traceback, IER is set to 1, and control is returned to the calling routine.

TIMING

Timing tests on sorted, random and reverse sorted arrays of various length (N) were done on the CYBER 170 Model 175 in 1979 May and are reported below:

Time (ms)

N	Sorted	Random	Sorted Reverse
32	.356	.743	.445
128	1.68	3.35	2.04
512	8.22	17.3	9.70
2048	39.8	85.6	46.4
8192	189.0	392.0	212.0

EXAMPLES

(1)

```
I=1
J=10
CALL ORDERF(A,I,J,TAG,IER)
```

Sorts A(1) through A(10) in ascending order and permutes TAG(1) through TAG(10) along with A.

(2)

```
I=5
J=25
CALL ORDERB(A,I,J,TAG,IER)
```

Sorts A(5) through A(25) in descending order and permutes TAG(5) through TAG(25) along with A.

REFERENCES

R.C. Singleton, An Efficient Algorithm for Sorting with Minimal Storage, Comm. ACM 12(3), 1969, pp. 185-187.

AUTHOR & REVISION HISTORY

Written by C.J. Johnson, January 1970 (as SORTAG/DESORT). Revised and name changed to ORDERF/ORDERB by P. Bumbulis, May 1979, to add an error flag to the argument list and to add comments and an error message to the code.

TITLE Set a Variable or an Array with a User-Defined Pattern

ENTRY CALL PRESET(S,I,P)

S is a real or integer variable, array element or array name indicating the starting address to be preset.

I is a positive integer input variable specifying the number of elements to be preset in S.

P is a real or integer input variable containing the pattern to be put in S.

Note that S and P need not be of the same type. If types are mixed, the preset value is the bit pattern in P. (See Example 3 below.)

EXIT Normal Exit: Control is returned to the calling routine with I words starting at the address of S set to the pattern in P.

Error Exit: If PRESET is called with a negative value of I, the message

```
***NUMBER OF WORDS TO PRESET LE 0 - FIRST PARAMETER
RETURNED INDEFINITE
```

is printed, the first word of S is set indefinite and control is returned to the calling routine.

TIMING The time required to preset an array of 100 elements is approximately 200 μs. (6600, December 1975)

EXAMPLES (1)

```
P=1.
I=100
CALL PRESET(A,I,P)
```

presets 100 words starting at the address of A (or A(1) if A is an array name) with the real constant 1.

(2) I=10
K=1
CALL PRESET(J,I,K)
presets 10 words starting at the address of J to the integer constant 1.

(3) I=10
K=2
CALL PRESET(X,I,K)
The 10 words starting at X will contain the integer representation of 2.

AUTHOR	D.G. Stewart	DATE	June 1974 Revised December 1975

TITLE

Reset Blocks of Core with a Pattern of Indefinites

INTRODUCTION

This routine sets each word of the specified block to the pattern

```
60000   00000   04004   00000 + IJKLMN
```

where IJKLMN is the octal address of that word.

Any variable within the specified block which is not subsequently defined will cause an error mode 4 abort if it is used in a floating point operation.

ENTRY

```
CALL RESET(A,B,I,J)
```

A real or integer input variable specifying the first word of the block

B real or integer input variable specifying the last word of the block or number of words in the block

I integer input control variable:

I=0 specifies that A is a variable name
I=1 specifies that A is a numeric address

J integer input control variable:

J=-1 specifies that B is a count
J=0 specifies that B is a variable name
J=1 specifies that B is a numeric address

EXIT

Control is returned to the calling routine after the specified block has been set. No error checking is done.

EXAMPLES

```
CALL RESET(F(11),13477B,0,1)
```

resets core from the address of F(11) to the address 13477B.

```
CALL RESET(X,100,0,-1)
```

resets 100 words starting at the address of X.

RESET

AUTHOR M.B. Carver DATE May 1972

DOCUMENTATION REVISED L.E. Evans DATE April 1978

2- 4 CHARACTER AND BIT STRING MANIPULATIONS

2-4 CHARACTER AND BIT STRING MANIPULATIONS

Introduction

FORTRAN contains standard language constructs for processing real, integer and logical data types, but it does not contain similar features for the direct handling of character and bit string data. There are, however, special tools available to manipulate these data types. These are:

i) Masking Operations provided within FTN
   - .AND.,.NOT. and .OR.[1]

ii) Intrinsic Functions provided with the FTN compiler
   - AND,OR,XOR,COMPL,SHIFT and MASK[2]

iii) Character and bit manipulation routines provided in this section of AELIB
   - LBYT,SBYT,PAKUNP,(BITS,CHAR,SHIFTB,SHIFTC)

Intrinsic vs External Subroutines

The FTN functions in (ii) as well as statement functions in general[3] are intrinsic functions. That is, the COMPASS code for the function is inserted directly into the user's program at compile time. An external subprogram, in comparison, is loaded separately from the program and control must pass to it and back when called. All AELIB subprograms are external subprograms.

The choice between intrinsic and external routines should depend on speed and storage requirements. Intrinsic routines are faster, but require a separate copy of the code for each use. External routines use only one copy but have the entry and return time overhead.

For a description of some FORTRAN statement functions for character and bit manipulation, see [4].

Usage Recommendations

i) Character and Bit String Transfers

While the use of FORTRAN intrinsic functions or statement functions would be best from an efficiency standpoint, for some applications, the ease of use of the AELIB routines is probably of greater importance. LBYT and SBYT are recommended for bit string extraction and replacement, respectively, operating on a single 60-bit word. PAKUNP is recommended for packing and unpacking a series of bit strings.

BITS and CHAR should not be used in new programs as they duplicate the capabilities of LBYT and SBYT and will probably be removed from the library.

ii) Shifting

The FORTRAN intrinsic function SHIFT should be used in preference to AELIB's SHIFTB or SHIFTC. (This AELIB routine was written before FORTRAN supplied a shift function. It is now no longer needed and will be removed from the library at a future revision.)

L.E. Evans
1981 May

REFERENCES

[1] FORTRAN Extended Version 4 Reference Manual, Revision E, Publication No. 60497800, Control Data Corporation, 1979, Chapter 2

[2] as [1], chapter 8

[3] as [1], chapter 7

[4] A Comprehensive List of FORTRAN Statement Functions for Character and Bit String Manipulations, an unpublished document available from Mathematics and Computation Branch Office.

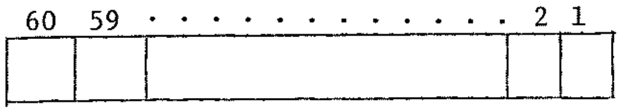
TITLE

Extract a Byte of any Length from anywhere in a Word.

INTRODUCTION

This function extracts a byte 1-60 bits long from a 60-bit word and stores this byte right justified in a 60-bit word with zero fill on the left.

The bit numbering convention used here is as follows:



ENTRY

I = LBYT(K,LENGTH,FROM)

K — integer input variable, the right most position of the byte to be extracted $1 \leq K \leq 60$

LENGTH — integer input variable, the length of the byte to be extracted $1 \leq LENGTH \leq 61-K$

FROM — integer, logical or real input variable, the word from which the byte will be taken. This word is not changed.

I — integer output variable in which the extracted byte is stored, right justified with zero fill.

EXIT

Control is returned to the calling routine following the byte transfer. No error checking is done.

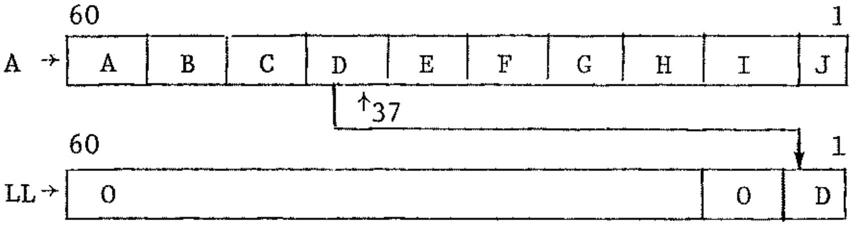
SPEED

18 μs (6600, December 1970)

EXAMPLE

Extract the fourth character from the left of word A and store it right justified in LL.

LL = LBYT(37,6,A)



LBYT

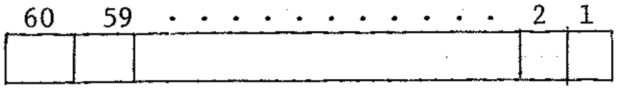
AUTHOR VIM Library Routine
Modified by J.L. Burton DATE December 1970

AUTHOR	VIM Library Routine Modified by J.L. Burton	DATE December 1970
DOCUMENTATION REVISED BY	L.E. Evans	April 1978

TITLE Store a Byte of Any Length anywhere in a Word.

INTRODUCTION This subroutine extracts a byte 1-60 bits long from the rightmost position of a 60 bit word and uses it to replace an equal length byte anywhere in another 60 bit word.

The bit numbering convention used is as follows:



ENTRY CALL SBYT(K,LENGTH,INTO,FROM)

K integer input variable, the rightmost position of the byte to be replaced. $1 \le K \le 60$

LENGTH integer input variable, the length of the byte to be replaced. $1 \le LENGTH \le 61-K$

INTO integer, real or logical output variable, the word in which the byte will be replaced.

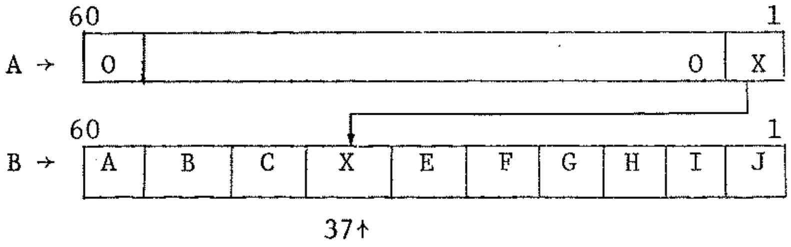
FROM integer, real or logical input variable, the word from which the byte is to be taken. The rightmost LENGTH bits are used.

EXIT Control is returned to the calling routine following the byte transfer. No error checking is done.

SPEED 22 μs (6600, December 1970)

EXAMPLE Store the rightmost 6 bits of word A in the fourth character position from the left of word B.

CALL SBYT(37,6,B,A)



SBYT

AUTHOR	VIM Library Routine Modified by J.L. Barton	DATE	December 1970

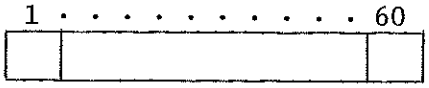
DOCUMENTATION REVISED BY L.E. Evans April 1978

TITLE Bit and Character Extraction or Replacement (this routine should not be used in new programs - see introduction to this section).

ENTRIES i) CALL BITS (ID,X,N,K,Y)

X,Y integer logical or real variables specifying the words to be used for extraction or replacement

ID,N,K integer input variables specifying bit extraction or replacement, using the bit numbering convention



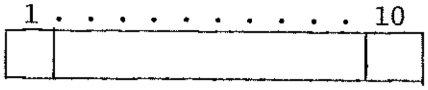
(Extraction) If ID=0, N bits of X, starting at bit K and counting to the right are stored in Y, left justified with zero fill. $(1 \leq N \leq 60, 1 \leq K \leq 60)$

(Replacement) If ID≠0, the N left most bits of Y, are stored in X starting at bit K and counting to the right. The remaining bits of X are unchanged. $(1 \leq N \leq 60, 1 \leq K \leq 60)$

ii) CALL CHAR(ID,X,N,K,Y)

X,Y integer, logical or real variables specifying the words to be used for extraction or replacement

ID,N,K integer input variables specifying character extraction or replacement using the character numbering convention:



(Extraction) If ID=0, N characters of X, starting at character K and counting to the right are stored in Y, left justified with blank fill.

(Replacement) If ID≠0, the N leftmost characters of Y are stored in word X, starting at character K and counting to the right. The remaining characters of X are unchanged.

EXIT Control is returned to the calling routine following the required transfer of bits. Note that end around transfers will take place if K+N-1 is greater than 60(10) for BITS (CHAR)

SPEED 20 μs for BITS, 25 μs for CHAR (6600 March 1969)

AUTHOR W.S. Chapman DATE March 1969

DOCUMENTATION REVISED BY L.E. Evans April 1978

TITLE Pack or Unpack Bit Strings of up to 60 Bits in Length

INTRODUCTION This routine copies consecutive bit strings from a source array into consecutive fields in a destination array ignoring word boundaries.

The starting position i_s (or i_d) of a source (or destination) string refers to the leftmost bit of the string.

Starting at bit position i_s in the first word of the source array, it copies n bit strings of length ℓ_s to destination fields of length ℓ_d in the destination array. The first destination field starts at bit position i_d in the first word of the destination array.

If $\ell_d=\ell_s$, the source strings are copied directly into the destination fields
If $\ell_d>\ell_s$, the source strings are stored right justified and zero filled to ℓ_d bits
If $\ell_d<\ell_s$, only the rightmost ℓ_d bits of the source string are copied into the destination field.

ENTRY CALL PAKUNP(SOURCE,LSS,IBS,N,DEST,LDF,IBD)

SOURCE real or integer variable or array containing the source strings

LSS integer input variable, the length in bits of the source strings $1\leq LSS\leq 60$

IBS integer input variable, the bit position, in the first word of SOURCE, at which the first source string starts. $0\leq IBS\leq 59$ (The bit numbering convention is

```
59 . . . . . . . . . . 0
|      FIRST WORD      |
```

N integer input variable, the number of bit strings to be copied.

DEST real or integer output variable or array which will contain the copied strings

LDF — integer input variable the length in bits of a destination field in which a source string will be stored $1 \le LDF \le 60$.

IBD — integer input variable, the bit position, in the first word of DEST, at which the first destination field starts. $0 \le IBD \le 59$

EXIT — The routine returns control to the calling program when the N source strings have been copied to the destination fields. Note that there is no checking of input parameters for validity.

SPEED — 20 to 35 μs per bit string copied (6600, December 1977). The speed depends on the number of 6600 words that have to be accessed, and the number of times bit strings cross word boundaries.

EXAMPLES

1) Unpack ten 6-bit characters from a single word into a 10 word array, right justified, zero filled.

```
CALL PAKUNP(SOURCE,6,59,10,DEST,60,59)
```

2) Pack the 10 characters that were unpacked in Example 1, back into the array SOURCE.

```
CALL PAKUNP(DEST,60,59,10,SOURCE,6,59)
```

3) Unpack twenty 8-bit fields from the array SOURCE, starting halfway through the first word of SOURCE, and store them in 12 bit fields starting with the right most 12-bit field in the first word of the destination array DEST.

```
CALL PAKUNP(SOURCE,8,29,20,DEST,12,11)
```

4) Extract the lower eight bits from the first ten 12-bit fields in the array SOURCE starting at the beginning of the first word, and pack them back into SOURCE starting at the beginning of the first word.

```
CALL PAKUNP(SOURCE,12,59,10,SOURCE,8,59)
```

Note that the above operation using the same array for source and destination, is possible because data compression is taking place.

AUTHOR G.N. Williams December 1977

TITLE Character and Bit Shifting by Subroutine Call. (This routine should not be used in new programs - see introduction to this section.)

ENTRIES i) CALL SHIFTB(X,N,Y)

X integer, logical or real input variable

Y integer, logical or real output variable

N integer input variable, $-59 \le N \le 59$.

If $N \le 0$, X is left shifted N bits (end around) into Y
If $N>0$, X is right shifted N bits (end off and sign extended) into Y

ii) CALL SHIFTC(X,N,Y)

X integer, logical or real input variable

Y integer, logical or real output variable

N integer input variable, $-10 \le N \le 10$.
If $N \le 0$, X is left shifted N characters (end around) into Y
If $N>0$, X is right shifted N characters (end off and blank fill) into Y.

EXIT Control is returned to the calling routine after the required shifting. No error checking is done.

SPEED 11 μs for SHIFTB (6600, April 1969)

For $N>0$, SHIFTC requires about 19 μs (6600, April 1969)
For $N \le 0$, SHIFTC requires about 13 μs (6600, April 1969)

AUTHOR W.S. Chapman DATE April 1969

DOCUMENTATION REVISED BY L.E. Evans April 1978

2- 5 SYSTEM FACILITIES

2-5 SYSTEM FACILITIES

All of the routines in this section interact to some degree with the NOS/BE operating system by

- retrieving system data such as the mainframe identifier
- calling system routines for some special purpose such as releasing a file, or
- extending current system software such as that for ECS addressing.

Because of their dependence on the operating system, any use of these routines should be clearly identified as a local feature in program documentation. For some of these routines, simplified versions for use at other computing centres are available and can be retrieved from the AELIB source tape using the UPDATE directive *DEFINE EXTERNAL. More details in these versions are supplied in the write-ups for the routines in which this feature is supported.

Dump Utilities

For any file, the routine DMPFET provides a listing of the File Information Table (FIT), the File Environment Table (FET) and the file buffer. Also note that the FTN library contains two routines (PDUMP and DUMP), to list specified sections of central memory[1].

Error Recovery

The FTN library routine RECOVR allows a user program to recover from execution errors by passing control to a user written subroutine[1]. The AELIB routine OFLOW recovers from a mode error by transferring control to a specified label in the user's program.

File Information and Manipulation

The AELIB routine RELESE may be used to return or unload files during the execution of a user's program. (More generally, extensive mass storage file utilities and FORTRAN Cyber Record Manager interface routines are provided in the FTN library and are described in [1].)

The AELIB subroutine ROUTE provides file routing capabilities from within a FORTRAN program in the same way as the ROUTE control statement does.

Field Length Management and ECS Addressing

Current CRNL support for dynamic field length management is described in AECL-6782[2]. We recommend that all users of dynamic memory management read this report.

The AELIB routines CHNGSCM and MEMSCM are for use with programs that have been compiled with FTN(STATIC). MEMSCM returns the current field length and CHNGSCM changes the field length to a specified value.

(CHNGSCM and MEMSCM are provided for compatability with the NOS/BE operating system at the previous 434 level and will be removed when all programs have been converted to the current 499 level.)

CHNGFX, CHNGEX and CHNGEC are more general field length management routines for fixed-length central memory blocks, expandable central memory blocks and ECS (respectively) and are recommended for new programs.

The routines CHNGFX and CHNGEX use the Common Memory Manager (CMM) [4] to do the actual work of managing field length. There are two versions of this product, an err-checking or safe version which performs extensive parameter checking, and a fast version which does not. Even though CHNGFX and CHNGEX validate parameters passed to them, it is advisable to use the safe version while testing a program using these two subroutines. To use the safe version, insert the following control statement into the load sequence (i.e. before the execute (LGO) control statement:

```
LIBLOAD(SYSLIB,CMMSAFE)
```

(See reference 4, page 2-13).

No action is required to use the fast version since this is the default version.

The routine MOVLUV transfers data to and from ECS resident arrays using the FTN library MOVLEV while allowing subscripts larger than 2^{17}-1 to be used for these arrays.

Return of Job Processing Parameters

The routines XTIME and TIMEIO retrieve central processor and input/output times respectively. MFID returns the mainframe identification of the machine on which a job is executing. JOBNAME returns the full seven character jobname of an executing program. USERAP returns accounting parameters for the job.

Program Execution Monitoring

SPY is a peripheral processor program which can monitor a program executing in the central processor. It uses a sampling technique to gather data on the amount of time spent executing in different sections of the code and writes this data on a disk file called DOSSIER in a form suitable for presentation as a histogram. The subroutines SPYON and SPYOFF control the execution of SPY. The central processor program PRNTSPY is used to analyze the data and print the resulting histogram.

Access to CCL Registers from FORTRAN Programs

The CDC CYBER Control Language (CCL) provides registers which can be set and tested using control statements [3]. The routines IGETCCL and SETCCL allow the CCL registers R1, R2, R3, and R1G to be read and set from within a FORTRAN program.

AELIB Utilities

i) Error Message Processing - Old (AELERTX,AELERR,AEFREQ)

Error processing for AELIB routines is discussed in detail in Part A, Appendix 5. AELERTX is a system text containing the text of error messages. AELERR issues error messages with traceback from library routines and should not be called from a user's program. A write-up for AELERR is provided for internal library use and to explain the associated coding in library source code. A simplified version of AELERR that prints error message numbers is also available as described in the write-up.

AEFREQ allows the user to control how often messages are printed by establishing a user defined printing interval.

ii) Error Message Processing - New (ALEROF, ALERON, ALERPR, ALERDM, TRACEB)

The routines in this group complement the printing of error messages (using PRINT statements) by providing user control over these messages and providing an automatic traceback report after the message has been printed.

ALEROF allows the user to disable all error messages for a particular routine and in some cases, specific messages for that routine. ALERON can be used to turn any or all messages back on. ALERPR is a logical function used by library routines to test the status of error messages, that is, whether they are to be printed or not. ALERDM prints a formatted dump of the status of error messages under user control to be used for debugging.

. TRACEB is called to print a traceback report following the printing of an error message.

Since these routines are fairly new (introduced in July 1979) the AELIB subroutines with which they can be used are listed at the end of the write-up for ALEROF.

iii) Error Message Processing from COMPASS Subroutines

The subroutine AEISER is provided to allow AELIB subroutines that have been written in COMPASS to issue messages using FTN output statements and to use the error message control described in (ii) above.

iv) Generation of AELIB Usage Statistics

Almost all AELIB routines have two calls to the routine LIBSTAT as their first executable statements. These calls count the number of times the AELIB routine is executed and maintain this information in an array in the user's field length. When the job terminates, special system dayfile messages are issued to record the number of times each routine has been called from that job. These messages are analyzed to produce AELIB usage statistics which are then used to direct library development.

LIBSTAT should not be called from a user's program. It is for internal AELIB use and a write-up is provided only to explain the calls in AELIB source code.

Since LIBSTAT was written to interface directly with our system it is imperative that copies of AELIB routines sent to other installations either have the LIBSTAT calls removed or are accompanied by a dummy LIBSTAT routine. Also, private copies of AELIB routines run on our system should have the LIBSTAT calls removed to avoid biasing AELIB usage statistics.

To get source code for a FORTRAN AELIB routine without LIBSTAT calls run the job specified in Part A, Section 5, adding the one extra UPDATE correction -

```
*DEFINE EXTERNAL
```

v) Setting Output Variable to Indefinites Following an AELIB Error

The routine SETBAD is provided to standardize the setting of output variables to indefinite following an AELIB error for which these variables cannot be assigned a meaningful value. (See Part 1, Appendix 5, for a description of the AELIB Error Processing Philosophy.)

L.E. Evans
1981 May

References

[1] FORTRAN Extended Version 4 Reference Manual, Revision E, Publication Number 60497800, Control Data Corporation, 1979, Chapter 8.

[2] Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System, M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart, Atomic Energy of Canada Limited report AECL-6782, February 1980.

[3] NOS/BE Version 1 Reference Manual, Revision K, Publication Number 60493800, Control Data Corporation, 1980, Chapter 5.

[4] Common Memory Manager Version 1 Reference Manual, Revision D, Publication Number 60499200, Control Data Corporation, 1980.

TITLE

Dump Contents of the File Information Table (FIT), the File Environment Table (FET) and the Buffer for a Specified File.

ENTRY

CALL DMPFET(ITN)

ITN is either logical unit number of a file (integer) (e.g. 2 ≡ TAPE2) or file name in display code (e.g. 5HTAPE2).

EXIT

The FIT is located and the contents are printed. If a buffer is present, then its contents are printed.

Possible error messages are:

*** FILE NAME NOT DECLARED - xxxxxx

The FIT for the file specified on the parameter ITN was not located. It probably specified a file that was not included on the PROGRAM card.

*** ZERO LENGTH BUFFER

The apparent length of the buffer is zero and, therefore, it will not be dumped.

*** BUFFER GREATER THAN 2001B

The buffer is too long to dump.

Error messages are followed by a traceback, then control is returned to the calling program.

ROUTINES CALLED

IGETFIT, a utility routine loaded with DMPFET, LIBSTAT, ALERPR and TRACEB from AELIB.

AUTHOR	L.E. Evans	DATE	May 1974
REVISED	C.J. Tanner	DATE	July 1979

TITLE

Return control to a user's program following a mode error.

ENTRIES

i) CALL OFLOW(I),RETURNS(lab)

sets a trap for mode errors

ii) CALL OFLOW

clears a previously set trap

I integer output variable which will be set to

$000MAAAAAA0000000000_8$

when a mode error is detected.

M specifies the number of the mode error.

AAAAAA specifies the octal program address where the mode error was detected.

lab a statement label in the user's program, for example, 150 or 2000, to which control will be transferred following the detection of a mode error.

ROUTINES CALLED

- RECOVR from the system library
- RECV.1 or RECV.2, utility routines loaded with RECOVR
- AEISER from AELIB

EXIT

After setting or clearing a mode error trap, control is returned to the calling program.

Once set, a mode error trap remains in force for the duration of the job step (i.e. until the next control card is processed) unless OFLOW is called to clear the trap or 50 mode errors are detected.

For each recovery mode, the dayfile message

MODE ERROR
JOB REPRIEVED

is issued (by RECOVR).

If 50 mode errors have been detected, the message

```
OFLOW - MORE THAN 50 MODE ERRORS
```

is printed, a traceback to the original OFLOW call is provided, and the last mode error will be restored to abort the job.

EXAMPLES

i) OFLOW call to return control to statement 150, following a mode error:

```
CALL OFLOW(I), RETURNS(150)
```

ii) If statement 150 prints I as

```
00020042240000000000
```

This means that an error mode 2 occurred at location 4224_8.

AUTHOR D. McPherson DATE February 1970
Revised August 1972

REVISED L.D. Hansen and D.G. Stewart June 1974

DOCUMENTATION REVISED

L.E. Evans April 1978

REVISED C.J. Tanner July 1979

TITLE Release a File

ENTRY I = RELESE(LFN,M)

LFN integer input variable specifying the file to be released. LFN may be either a logical unit number or a logical file name left justified with zero fill. (Examples: 5LTAPE5, 5)

M integer input control variable.

M=0 or absent causes the file to be returned (i.e. CLOSE RETURN function on the file)

M≠0 causes the file to be unloaded (i.e. CLOSE UNLOAD function on the file)

I integer output variable in which the function value is stored. It will be 0 if the file was unloaded or returned successfully and -1 otherwise.

Note: RELESE must be declared INTEGER in the calling program.

EXIT Normal Return

I is set to 0 and a message is issued to the dayfile as follows:

RELESE ... lfn

where lfn is the name of the file returned/unloaded.

Error Messages

If an error is detected a message will be issued followed by a traceback. Control will be returned to the calling routine with I set to -1. The following error messages can be issued:

*** LOGICAL UNIT NUMBER nn OUT OF RANGE

occurs when LFN is an integer. Indicates that LFN ≤ 0 or LFN > 99.

*** CANNOT CLOSE A MASS STORAGE FILE

when releasing a mass storage file CLOSMS must be called before RELESE is called.

*** ERROR IN RETURN - SEE DAYFILE

the dayfile contains a message from the operating system describing an error that occurred when attempting to return/unload the file.

ROUTINES CALLED

AEISER from AELIB. IGETFIT (to get FIT address), AEFLFN (to process file name), AECLOS (to close a file) and AERETR (to return a file) are all utility routines loaded with RELESE.

EXAMPLE The following program uses TAPE5 as a scratch file, then releases it so that it can be used again for something completely different.

```
      PROGRAM XYZ (INPUT, OUTPUT, TAPE5)
C
      INTEGER RELESE
      DIMENSION A(10)

C  PUT SOME VALUES INTO A.
      DO 100 I=1,10
      A(I) = I
100   CONTINUE

C  WRITE A ONTO TAPE5 A FEW TIMES
      DO 110 I=1,100
      WRITE (5) A
110   CONTINUE

C  CALL A SUBROUTINE TO DO THE FIRST PART OF THIS PROGRAM.
C  THIS SUBROUTINE WILL READ THE DATA ON TAPE5.
      CALL MAINPR

C  NOW WE ARE FINISHED THE CURRENT CONTENTS OF TAPE5. CALL
C  RELESE TO GET RID OF THEM.
      I = RELESE(5,0)
      IF (I.NE.0) STOP "ERRORS"

C  NOW TAPE5 IS EMPTY. CALL THE SUBROUTINE TO DO THE SECOND
C  PART OF THIS PROGRAM. IT WILL WRITE SOMETHING ONTO TAPE5
C  TO BE SAVED FOR LATER.
      CALL SECND
      WRITE (5,1)
1     FORMAT (*  END OF DATA*)
      END
```

AUTHOR	L.D. Hansen	DATE	July 1974
DOCUMENTATION REVISED			
	L.E. Evans	DATE	April 1978
REVISED	C.J. Tanner	DATE	June 1979

TITLE

Route a File to an Output Queue. This subroutine performs the same function as the ROUTE control statement described in [1].

ENTRY

CALL ROUTE(LFN,IER,PARAM1,VALUE1,...,PARAMn,VALUEn)

where

LFN name of file to be routed; either logical unit number (integer), or name of file in left-justified display code (e.g. 5HTAPE5).

IER error code returned.

PARAM optional parameter (see table below). The parameter name is given in display code, left-adjusted.

VALUE value of the parameter, display code, left-adjusted when the value is a mnemonic, otherwise an integer.

Param Value

DEF =0, release file now (default).
=1, release file at end of job.

DC =dc, file disposition. Allowed values on both NOS/BE and NOS are:

SC	evict the file
PR	print
PU	punch the file
PB	punch system binary
SB	punch system binary (same as PB)
P8	punch 80-column cards
PL	plot on any plotter
IN	place file in the input queue

Allowed values on NOS/BE only are:

PM	print with special instructions
PV	punch with special instructions
PW	wide plot
PX	plot with special instructions
PY	wide plot with special instructions

HN	Harris plot, narrow
HW	Harris plot, wide
HX	Harris plot with special instructions
HY	Harris plot, wide, with special instructions
GR	device independent file

Additional values for NOS/BE only that apply only to files transmitted to keyboard terminals via the 3300 are:

P0	transmit in 8-bit mode, no parity
P1	3300 edit files
P5	print a file produced in the ASCII mode of CREATE
P6	transmit in 6-bit binary mode with bit 6 set, plus parity
P7	transmit in 7-bit mode plus parity

EC =ec, external characteristics, used for punch files:

Omitted	punch 026 mode
026 or 026	punch 026 mode
029 or 029	punch 029 mode
SB	punch system binary
80COL	punch 80-column binary

FID (NOS/BE only) =fid, file name while the file is in the OUTPUT queue.

*, first five characters of the file name are the same as on the job card. Two unique sequence numbers different from the job sequence numbers, are added in the sixth and seventh positions.

fffff, first five characters of the file name are fffff. Characters 6 and 7 are the same sequence characters as in the job name.

*fffff, equivalent to fffff except that two unique sequence numbers different from the job sequence numbers are added as characters 6 and 7.

REP =n, number of extra copies to be produced. $n \leq 31$.

ST =st, destination station id.

TID =tid, terminal identification.

UN =un, user name/number (NOS only).

EXIT If no errors are encountered, IER is set to zero and the specified routing will be performed after flushing buffer for file (if necessary).

If an error is encountered, IER will be set non zero and an error message issued.

Value of IER	Error Message
1	INVALID LFN - DSP
2	CANT ROUTE NON ALLOCATABLE EQP File to be routed cannot be on a medium such as magnetic tape.
3	CANT ROUTE PERM FILE NOS: This error includes trying to route the primary file.
4	NO PERMISSION TO ROUTE THIS FILE See your site consultant. NOS: This error could indicate an output file limit.
6	IMMEDIATE ROUTING - NO FILE - IGNORED DEF=0 but no file exists.
7	INVALID DISPOSITION CODE - ROUTING IGNORED
10	INVALID FID - ROUTING IGNORED
11	DSP ABORTED BY SYSTEM Unless there is further explanation in the dayfile (time limit, etc.) see your site consultant.
12	DSP PARAMETER OUTSIDE FL Could mean code in ROUTE overwritten.

17	DSP COMPLETE BIT ALREADY SET See your site consultant.
21	TID NOT ALPHANUMERIC - ROUTING IGNORED
22	INVALID LINK TYPE - ROUTING IGNORED Could indicate system problem, see site consultant.
24 (NOS/BE only)	FILE WILL BE COPIED TO QUEUE DEVICE Queue files are kept on a separate device. This file will be copied to the correct device before being put into the requested queue. See [1] for further details.
99	UNKNOWN ERROR CODE - nn DSP has returned an error code that was not expected. See page 7-80 of the NOS/BE Reference Manual for a table of DSP error codes, or your site consultant.
101	REPEAT COUNT nn GREATER THAN 37B
102	UNKNOWN EXTERNAL CHARACTERISTICS XXX
103	UNKNOWN PARAMETER XXX
104	DUPLICATE PARAMETER XXX Same parameter given more than once.

The appropriate error message and a traceback will be printed and control returned to the calling routine. No routing will be done.

ROUTINES CALLED

- AEFLFN, IGETFIT, AECLOS from RELESE; DSP - PP program; and RTINIT - initialize ROUTE; RTSTOR - store a parameter; RTERR - issue error message; RTBLNK - remove trailing blanks, utility routines loaded with ROUTE.

- ALERPR, LIBSTAT and TRACEB from AELIB.

COMMON BLOCKS

RTCOMM - internal storage for ROUTE.

EXAMPLE

(a) CALL ROUTE(OUTPUT,IER,3LDEF,1,2LDC,2LPM)

will set special disposition for the output file when it is released at the end of the job.

Note: When setting DC=PM (print) or PX (plot) instructions telling the operator how to process the file must be included with the job.

(b) CALL ROUTE(1,IER,2HDC,2HPR,3HFID,5HABCDE)

will route the file TAPE1 to the printer (at the terminal from which this job came). The name of the file in the queue and on the page banner will be ABCDExx where xx are the sequence characters of the job.

REFERENCE

[1] NOS/BE Reference Manual, CDC Publication #60493800, Revision K, 1980, page 4-83, or
NOS Reference Manual, Volume 1, CDC Publication #60435400, Revision M, 1980, page 1-7-34.

AUTHOR C.J. Tanner

DATE September 1979
REVISED June 1983

TITLE

Field length control for programs running in static mode (i.e. compiled with FTN(STATIC,...)). Dynamic memory management is supported by the AELIB routines CHNGFX, CHNGEX and described in detail in [1]. CHNGSCM is supported only for those programs that have not been modified to use CHNGFX,CHNGEX and will be removed from AELIB when no longer required.

ENTRY

CALL CHNGSCM(IFL)

IFL Integer input variable, the desired field length in words.

EXIT

Normal Exit: The field length for this job is set to the value of IFL.

Error Exits: If the value of IFL is found to be illegal and if an OUTPUT file exists, the fatal error message

```
***ILLEGAL FIELD LENGTH = (IFL)B.  PARAMETER IS EITHER
NEG, REAL, TOO LARGE OR LESS THAN CHNGSCM MEMORY LOCA-
TION.
```

along with trace information is printed. If this value of IFL is such that honoring the CHNGSCM demand would result in a field length greater than the limit for the job, then the fatal error message

```
CANT MEM ABOVE JOBCARD CM
```

is produced in the dayfile.

EXAMPLE

CALL CHNGSCM(30000B)

changes the field length for this job to 30000 octal words.

ROUTINES CALLED

AELERR from AELIB.

COMMON BLOCKS USED

AELERCM

REFERENCE

[1] Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System, M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart, Atomic Energy of Canada Limited report AECL-6782, February 1980.

CHNGSCM

AUTHOR	L.D. Hansen	DATE	June 1975
REVISED	L.D. Hansen	DATE	December 1975
	D.G. Stewart		December 1979

TITLE

Retrieve current job field length for programs running in static mode (i.e. compiled with FTN(STATIC,...)). Dynamic memory management is supported by the AELIB routines CHNGFX, CHNGEX and described in detail in [1]. MEMSCM is supported only for those programs that have not been modified to use CHNGFX, CHNGEX and will be removed from AELIB when no longer required.

ENTRY

I = MEMSCM(DUMMY)

DUMMY	any type of variable, not used by MEMSCM but required by FTN to recognize MEMSCM as a function
I	integer output variable in which the field length is stored on return from MEMSCM.

EXIT

Control returns to the calling routine after the current job field length is stored in I.

REFERENCE

[1] Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System, M.B. Carver, C.J. Tanner, G.L. Klawitter, and D.G. Stewart, Atomic Energy of Canada Limited report AECL-6782, February 1980.

AUTHOR	L.D. Hansen	DATE	June 1975
REVISED	D.G. Stewart	DATE	December 1979

TITLE Dynamic Memory Management Routine to Allocate and Release Fixed Length Storage Blocks.

INTRODUCTION With the installation of the NOS/BE 1.3 Operating System, and the use of capsules by CRM, etc., a new dynamic field length management system was implemented to replace the previous MEMSCM and CHNGSCM. The routine CHNGFX is part of this system that is fully described in AECL-6782, Dynamic Allocation of core on the CRNL NOS/BE 1.3 System, by M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart.

ENTRY L=CHNGFX(Z,INC,IFW,ISP,IDO)

Z Real, input, blank common array element used as the base position for dynamic memory management, usually the first word in blank common. The user must declare Z as an array in blank common in the calling routine. If Z is the first word in blank common COMMON Z(1) is sufficient.

INC Integer, input variable specifying the number of words of new working storage required.

IFW Integer variable used as input as defined in the table below. If CHNGFX is called to allocate storage and is successful, IFW will return the index, in array Z, of the first word of the new storage block. The value of IFW is relative to Z hence the user must use FTN routine LOCF to determine the address of Z(IFW).

ISP Integer, output variable specifying the number of words in the largest block that can be assigned for working storage.

IDO Integer, input variable specifying with IFW the operations CHNGFX is to perform.

Allocate Storage

IDO=0, IFW=0 Allocate a fixed block; i.e. allocate a block above Highest High Address (HHA) with an unrestricted lifetime.

IDO>0, IFW=0 Allocate a temporary block; i.e. allocate a block of storage preferably below HHA (although it may not be). This block must be released before the next overlay change.

Free Existing Storage and Allocate New Storage

IDO≥0, IFW>0 Free fixed or temporary block Z(IFW) before allocating new storage block.

Free Storage

IDO<0, IFW>0 Free fixed or temporary block Z(IFW).

IDO<0, IFW=0 Free all temporary blocks.

L Logical variable reporting to the user the success or failure of CHNGFX.

ROUTINES CALLED

- CMMALC, CMMFRE, CMMFTB, ICMMGFS, ICMMAGR, CMMFGR Common Memory Management (CMM) routines loaded with CHNGFX (see Appendix 4 for deck location of source code for these modules).

- ALERPR, TRACEB and LIBSTAT from AELIB.

COMMON BLOCKS

Z must be declared as an array in blank common as described above.

EXITS

NORMAL EXIT

If CHNGFX is able to allocate and/or free the requested storage blocks, control is returned to the calling routine with L=.TRUE. and ISP reporting the new space available.

ERROR EXITS

(1) If CHNGFX is called to allocate more storage than is available, i.e. INC>ISP, control is returned to the calling program with L=.FALSE. and ISP reporting the current available space.

(2) If CHNGFX is called with INC≤0, the error message

***INPUT PARAMETER VIOLATION. INC≤0 IS INVALID.
INC=

is printed with traceback. Control is returned to the calling routine with L=.FALSE. and ISP reporting the currently available storage.

(3) An attempt to load an overlay after one or several temporary blocks have been acquired without freeing all temporary blocks by calling CHNGFX with IDO<0 and IFW=0 results in a FTN - FATAL ERROR 52 and an appropriate CMM error message. The user's program is aborted in this case and in other cases in which errors are detected by CMM.

EXAMPLE

```
      PROGRAM TCHNGFX(INPUT,OUTPUT)
C
C  PURPOSE : TO ACQUIRE A TEMPORARY BLOCK AND THEN RELEASE IT.
C
C  IDO.EQ.0 WITH IFW.EQ.0 SIGNIFIES THAT THE ACQUIRED BLOCK IS TO BE A
C                         TEMPORARY BLOCK IE. THE BLOCK MUST BE RELEASED
C                         BEFORE THE NEXT OVERLAY CHANGE.
C  IDO.EQ.-1 WITH IFW.GE.0 SIGNIFIES THAT TEMPORARY BLOCK Z(IFW) IS TO
C                          BE RELEASED.
C
C  SINCE THE LOCATION OF BLOCKS ACQUIRED VIA DYNAMIC ALLOCATION OF CORE
C  IS NORMALLY DESIRED TO BE RELATIVE TO BLANK COMMON, THE FIRST WORD
C  OF THE BLOCK ACQUIRED BY CALLING CHNGFX IS RELATIVE TO THE FIRST
C  WORD IN A BLANK COMMON BLOCK Z.
C
      COMMON Z(100)
      LOGICAL CHNGFX,TEST
C
C     PRINT OUTPUT HEADER.
C
      PRINT 200
C
C     ******  ACQUIRE A BLOCK OF LENGTH INC=500, SUBSEQUENT NEW IFW AND
C             CHECK THE SUCCESS OF SAID ACQUISITION.
C
      IFW = 0
      INC = 500
      IDO = 1
      IF(CHNGFX(Z,INC,IFW,ISP,IDO)) 190,180
C
C        TEST=.FALSE.  OUTPUT APPROPRIATE MESSAGE.  TERMINATE EXECUTION.
C
  180    PRINT 220,INC,ISP
         STOP
```

```
C
C         TEST=.TRUE.  OUTPUT APPROPRIATE MESSAGE.  CONTINUE.
C
  190     PRINT 210,INC,IFW,LOCF(Z(IFW)),ISP
C
C         ******  FREE TEMPORARY BLOCK PREVIOUSLY ACQUIRED.
C
          IDO = -1
          TEST = CHNGFX(Z,INC,IFW,ISP,IDO)
C
C         OUTPUT APPROPRIATE MESSAGE AS DICTATED BY THE VALUE OF TEST.
C
          IF(TEST) PRINT 230,IFW,ISP
          IF(.NOT.TEST) PRINT 240
C
C
  200 FORMAT("1",///11X,"**********                        CHNGFX TEST
     ,      **********",//11X,"(OUTPUT IS EXPRESSED IN OCTAL REPRESENTATION
     ,)")
  210 FORMAT("0",10X,"A TEMPORARY BLOCK OF LENGTH ",O5," HAS BEEN ACQUIR
     ,ED",/15X,"FIRST WORD OF ACQUIRED BLOCK (RELATIVE TO Z(1)) = ",O6,/
     ,15X,"ADDRESS OF SAID FIRST WORD = ",O6,/15X,"AVAILABLE SPACE REMAI
     ,NING = ",O6)
  220 FORMAT("0",10X,"***  CHNGFX=.FALSE.   NO BLOCK ACQUIRED.   INC=",O
     ,5," > ISP=",O6)
  230 FORMAT("0",10X,"TEMPORARY BLOCK Z(",O6,") HAS BEEN FREED TO CMM",/
     ,15X,"AVAILABLE SPACE REMAINING = ",O6,//11X,"**********
     ,         END OF TEST              **********")
  240 FORMAT("0",10X,"***  CHNGFX=.FALSE. NO BLOCKS FREED.")
      STOP
      END
```

```
**********                        CHNGFX TEST          **********

(OUTPUT IS EXPRESSED IN OCTAL REPRESENTATION)

A TEMPORARY BLOCK OF LENGTH 00764 HAS BEEN ACQUIRED
    FIRST WORD OF ACQUIRED BLOCK (RELATIVE TO Z(1)) = 001414
    ADDRESS OF SAID FIRST WORD = 014700
    AVAILABLE SPACE REMAINING = 304031

TEMPORARY BLOCK Z(001414) HAS BEEN FREED TO CMM
    AVAILABLE SPACE REMAINING = 305017

**********                        END OF TEST          **********
```

REFERENCE Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System, M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart Atomic Energy of Canada Limited report AECL-6782, February 1980.

AUTHORS M.B. Carver and G.L. Klawitter DATE December 1979

TITLE

Dynamic Memory Management Routine to Allocate, Release, Shrink and/or Terminate the Variable Status of an Expandable Storage Block.

INTRODUCTION

With the installation of the NOS/BE 1.3 Operating System, and the use of capsules by CRM, etc., a new dynamic field length management system was implemented to replace the previous MEMSCM and CHNGSCM. The routine CHNGEX is part of this system that is fully described in AECL-6782, Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System, by M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart.

ENTRY

L=CHNGEX(Z,INC,IFW,ISP,IDO)

Z Real, input, blank-common array element used as the base position dynamic memory management, usually the first word in blank common. The user must declare Z as an array in blank common in the calling routine. If Z is the first word in blank common COMMON Z(1) is sufficient.

INC Integer, input variable specifying the number of words of new working storage required.

IFW Integer input and output variable. In cases (a) and (b)below, it is used as a control variable to specify the type of allocation to be performed. If this allocation is successful, IFW will return the index, in array Z, of the first word of the new storage block. (The value of IFW is relative to Z hence the user must use FTN routine LOCF to determine the address of Z(IFW).) In cases (c) and (d) below, IFW is used as an input variable only to define the location of the block being altered.

ISP Integer, output variable specifying the number of words in the largest block that can be assigned for working storage.

IDO Integer, input variable specifying with IFW and INC the operating CHNGEX is to perform.

(a) Allocate an Expandable Block

IDO=0, IFW≤0, INC>0

Allocate an expandable block. Note that until the expandable status of this block is terminated by a further call of CHNGEX defined below, the system cannot assign further blocks or load weak externals. Therefore, the expandable status should be terminated as soon as possible.

(b) Free Existing Storage and Allocate New Storage

IDO=0, IFW>0, INC>0

Free block Z(IFW) and then allocate an expandable block.

(c) Free an Allocated Block

IDO<0, Free block Z(IFW)

(d) Dynamically Alter Allocated Block

IDO>0, INC>0

Expand block Z(IFW) by INC. Also, if IDO>1 terminate expandable status of block. (If IDO=1, leave block in expandable status.)

IDO>0, INC<0

Shrink block Z(IFW) by INC. Also if IDO>1, terminate expandable status of block. (If IDO=1, leave block in expandable status.)

IDO>0, INC=0

Terminate the expandable status of block Z(IFW).

L Logical variable reporting to the user the success or failure of CHNGEX.

ROUTINES CALLED - CMMFRE, ICMMGFS, ICMMALF, CMMCSF, CMMSLF, CMMGLF Common Memory Management (CMM) routines loaded with CHNGEX (see Appendix 4 for deck location of source code for these modules).

- ALEPRR, TRACEB and LIBSTAT from AELIB.

COMMON BLOCKS

Z must be declared as an array in blank common as described above.

EXITS

NORMAL EXIT

If CHNGEX is able to acquire, expand, shirnk and/or terminate the variable status of a fixed, expandable block, control is returned to the calling routine with L=.TRUE. and ISP reporting the new space available.

ERROR EXITS

(1) If CHNGEX is called to allocate or expand a fixed, expandable block to a size greater than the available storage, i.e. INC>ISP, control is returned to the calling program with L=.FALSE. and ISP reporting the current available space.

(2) If CHNGEX is called with IDO=0 and INC≤0, the error message

***INPUT PARAMETER VIOLATION. INC≤0 IS INVALID,
INC=

is printed with traceback. Control is returned to the calling routine with L=.FALSE. and ISP reporting the current available storage.

(3) An attempt to shrink an expandable block below FWA (first word address) results in a FTN - FATAL ERROR 52 and an appropriate CMM error message. The user's program is aborted in this case and in other cases in which errors are detected by CMM.

EXAMPLE

```
      PROGRAM TCHNGEX(INPUT,OUTPUT)
C
C  PURPOSE : TO ACQUIRE AN EXPANDABLE BLOCK AND THEN EXPAND AND
C            TERMINATE THE VARIABLE STATUS OF SAID BLOCK.
C
C  IDO.EQ.0 WITH IFW.LE.0 INDICATES THAT A NEW IFW FOR FIXED, EXPAND-
C                         ABLE BLOCK Z(IFW) IS TO BE OBTAINED.
C  IDO.GT.1 WITH INC.GT.0 INDICATES THAT BLOCK Z(IFW) IS TO BE EXPANDED
C                         BY INC AND THEN FIXED IE. THE VARIABLE STATUS
C                         OF BLOCK Z(IFW) IS TO BE TERMINATED.
C
C  SINCE THE LOCATION OF BLOCKS ACQUIRED VIA DYNAMIC ALLOCATION OF CORE
C  IS NORMALLY DESIRED TO BE RELATIVE TO BLANK COMMON, THE FIRST WORD
C  OF THE BLOCK ACQUIRED BY CALLING CHNGEX IS RELATIVE TO THE FIRST
C  WORD IN A BLANK COMMON BLOCK Z.
C
C
      COMMON Z(100)
      LOGICAL CHNGEX,TEST
C
C     PRINT OUTPUT HEADER.
C
      PRINT 100
C
C     ******  ACQUIRE NEW IFW FOR EXPANDABLE BLOCK Z(IFW) AND CHECK
C             THE SUCCESS OF SAID ACQUISITION.
C
      IFW = 0
      INC = 1000
      IDO = 0
      IF(CHNGEX(Z,INC,IFW,ISP,IDO)) 90,80
C
C        TEST=.FALSE.  OUTPUT APPROPRIATE MESSAGE. TERMINATE EXECUTION.
C
   80    PRINT 110,INC,ISP
         STOP
C
C        ******  TEST=.TRUE.
C        ******  EXPAND BLOCK Z(IFW) BY INC=1000 AND FIX IT.
C
   90    IDO = 2
         TEST = CHNGEX(Z,INC,IFW,ISP,IDO)
C
C        OUTPUT APPROPRIATE MESSAGE AS DICTATED BY THE VALUE OF TEST.
C
         IF(TEST) PRINT 120,INC,IFW,LOCF(Z(IFW)),ISP
         IF(.NOT.TEST) PRINT 130,IFW,INC,ISP
C
  100 FORMAT("1",10X,"**********                          CHNGEX TEST
     ,**********",//11X,"(OUTPUT IS EXPRESSED IN OCTAL REPRESENTATION)")
  110 FORMAT("0",10X,"***  CHNGEX=.FALSE.   NEW IFW FOR EXPANDABLE BLOCK
     , Z(IFW) NOT AVAILABLE.  INC=",O5," > ISP=",O6)
  120 FORMAT("0",10X,"AN EXPANDABLE BLOCK OF LENGTH ",O5," HAS BEEN ACQU
     ,IRED AND FIXED",//15X,"FIRST WORD OF ACQUIRED BLOCK (RELATIVE TO Z
     ,(1)) = ",O6,/15X,"ADDRESS OF SAID FIRST WORD = ",O6,/15X,"AVAILABL
     ,E SPACE REMAINING = ",O6,//11X,"**********                 END OF
     , TEST          **********")
  130 FORMAT("0",10X,"***  CHNGEX=.FALSE.   BLOCK Z(",O5,") NOT EXPANDED
     ,. INC=",O5," > ISP=",O6)
      END
```

```
**********                CHNGEX TEST                **********
(OUTPUT IS EXPRESSED IN OCTAL REPRESENTATION)
AN EXPANDABLE BLOCK OF LENGTH 01750 HAS BEEN ACQUIRED AND FIXED
    FIRST WORD OF ACQUIRED BLOCK (RELATIVE TO Z(1)) = 001413
    ADDRESS OF SAID FIRST WORD = 014455
    AVAILABLE SPACE REMAINING = 301321
**********                END OF  TEST               **********
```

REFERENCE Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System
M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart,
Atomic Energy of Canada Limited report AECL-6782, February
1980.

AUTHOR M.B. Carver and G.L. Klawitter DATE December 1979

TITLE Read or Write ECS Addresses Greater than 2^{17}

INTRODUCTION The FORTRAN library routine MOVLEV is used to read and write Extended Care Storage (ECS). The entry to this routine is

```
CALL MOVLEV(A,B,N)
```

where A is the first word of the information to be moved.

B is the first word in which the data is to be stored.

and N is the number of words to be moved.

However, assuming

A is resident in ECS and B in central memory, then

```
CALL MOVLEV(A(J),B,N)
```

will work only if $J \leq 2^{17}-1$ (131071_{10}).

The FORTRAN specifications dictate that subscripts should not exceed this number, but while this restriction is suitable for CM addresses, there is no reason why it should apply to ECS.

The routine MOVLUV has been provided to remove this restriction from ECS addresses. The user computes J and passes it separately to MOVLUV. MOVLUV computes the address A(J) or B(J) correctly and then calls MOVLEV. So,

```
     CALL MOVLUV(A,B,N,J) translates to

     CALL MOVLEV(A,B(J),N)  if B is ECS resident and
to   CALL MOVLEV(A(J),B,N)  if A is ECS resident.
```

ENTRY CALL MOVLUV(A,B,N,J)

A input variable or array element, the first word of the information to be moved

B output variable or array element, the first word in which the data is to be stored

N	integer input variable, the number of words to be moved
J	integer input variable, the index to be applied to the ECS base address, either A or B.

EXIT Control returns to the calling routine after the transfer is complete.

ROUTINES CALLED MOVLEV from the FORTRAN library.

AUTHOR	M.B. Carver	DATE	December 1975
DOCUMENTATION REVISED	L.E. Evans		April 1978

TITLE

Routine to Dynamically Allocate Extended Core Storage (ECS) as Required by the User.

INTRODUCTION

With the installation of the NOS/BE 1.3 operating system, a new dynamic field length management system was implemented. For a full description of dynamic field length management, see AECL-6782, Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System, by M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart. The previous field length management routines, MEMSCM and CHNGSCM, manipulated central memory storage only. The routine CHNGEC has been included in the new system to provide dynamic allocation of Extended Core Storage (ECS) as well.

ENTRY

L=CHNGEC(NECS,IAVL,ISP)

NECS Integer variable used as input to specify the amount of ECS required and as output to report the amount of ECS allocated. Note that the amount of ECS allocated is NECS rounded to the next highest 1000_8 words.

IAVL Integer, output variable returning total amount of ECS as specified on the job card.

ISP Integer, output variable specifying the remaining ECS space available.

L Logical variable reporting to the user the success (.TRUE.) or failure (.FALSE.) of CHNGEC.

ROUTINES CALLED

CHNGECS, ICMMGEC and MEMECS Common Memory Management (CMM) routines loaded with CHNGEC and ALERPR, TRACEB and LIBSTAT from AELIB.

EXITS

Normal Exit:

If CHNGEC is able to allocate the requested ECS, control is returned to the calling routine with L=.TRUE. and NECS reporting the amount of ECS allocated, IAVL the total amount available and ISP the amount remaining.

Error Exits:

(a) CHNGEC is called with NECS less than 0, unless turned off by a call to ALEROF, the error message

***INPUT PARAMETER VIOLATION. NECS NEGATIVE, NECS=

is printed with traceback. Control is returned to the calling program with L=.FALSE. and NECS reporting the current ECS available.

(b) If NECS is greater than the amount of ECS specified on the job card, control is returned to the calling routine with L=.FALSE. and NECS reporting the current ECS available.

EXAMPLE The following program calls CHNGEC to allocate three blocks of ECS of length 500_8, 2000_8, $25,000_8$ respectively. Note that in the first case, 1000_8 words will be allocated.

```
      PROGRAM ECSS(OUTPUT,INPUT)
C
C *********************************************************************
C ******                                                         ******
C ******        EXAMPLE PROGRAM TESTING CHNGEC    2-05-36        ******
C ******                                                         ******
C ******         THREE REQUESTS FOR DIFFERENT AMOUNTS OF ECS     ******
C ******     1   500B WILL BE ROUNDED TO 1000B AND RETURN TRUE   ******
C ******     2   2000B WILL RETURN TRUE                          ******
C ******     3   25000B GT AMOUNT ON JOB CARD RETURNS FALSE      ******
C ******                                                         ******
C ******         AUTHOR D.G. STEWART    JANUARY 1980             ******
C ******                                                         ******
C *********************************************************************
C
      COMMON/ECS/ECS(10)
      LEVEL 3,ECS
      INTEGER NE(3),NANS(3),SUC(2)
      LOGICAL CHNGEC
```

```
C
      DATA NE/500B,2000B,25000B/
      DATA NANS/1000B,2000B,2000B/
      DATA ISUCC,SUC/1,2R  ,2RUN/
      DATA ITRUE,IFALSE/5HTRUE ,5HFALSE/
C
C      PRINT OUTPUT HEADER
C
       PRINT 100
C
       DO 10 I=1,3
       PRINT 110,I
       NECS = NE(I)
       PRINT 130,NECS
       IF(CHNGEC(NECS,IAVL,ISP)) 20,30
 20    CONTINUE
       PRINT 150,ITRUE,NECS,ISP,IAVL
       GO TO 40
C
 30   CONTINUE
      PRINT 150,IFALSE,NECS,ISP,IAVL
C
 40    CONTINUE
C
C      CHECK TO MAKE SURE TEST IS SUCCESSFUL
       IF(NANS(I).NE.NECS) ISUCC=2
C
 10    CONTINUE
       PRINT 160,SUC(ISUCC)
C
 100  FORMAT("1",10X,"************      CHNGEC TEST   2-05-36
     1***********",//21X,"OUTPUT IS EXPRESSED IN OCTAL REPRESENTATION ")
 110  FORMAT(///,10X,"********************* CASE ",I1," ****************
     1****"//)
 130     FORMAT(* ASKING FOR *,O6,*B ECS*//)
 150   FORMAT(* FUNCTION CHNGEC RETURNS *,A5,* WE NOW HAVE *,O6,*B ECS*/,
     1* WITH *,O6,*B ECS REMAINING OF A TOTAL ECS *,O6,*B AVAILABLE*)
 160  FORMAT(///,10X,"*********** CHNGEC TEST WAS ",R2,"SUCCESSFUL *****
     1*****")
        END
```

```
            ************        CHNGEC TEST    2-05-36           ***********
                           OUTPUT IS EXPRESSED IN OCTAL REPRESENTATION


           ******************** CASE 1 ********************

   ASKING FOR 000500B ECS

   FUNCTION CHNGEC RETURNS TRUE  WE NOW HAVE 001000B ECS
   WITH 017000B ECS REMAINING OF A TOTAL ECS 020000B AVAILABLE


           ******************** CASE 2 ********************

   ASKING FOR 002000B ECS

   FUNCTION CHNGEC RETURNS TRUE  WE NOW HAVE 002000B ECS
   WITH 016000B ECS REMAINING OF A TOTAL ECS 020000B AVAILABLE


           ******************** CASE 3 ********************

   ASKING FOR 025000B ECS

   FUNCTION CHNGEC RETURNS FALSE WE NOW HAVE 002000B ECS
   WITH 016000B ECS REMAINING OF A TOTAL ECS 020000B AVAILABLE


           *********** CHNGEC TEST WAS   SUCCESSFUL *********
```

REFERENCE "Dynamic Allocation of Core on the CRNL NOS/BE 1.3 System", by M.B. Carver, C.J. Tanner, G.L. Klawitter and D.G. Stewart, AECL-6782, February 1980.

AUTHOR D.G. Stewart DATE January 1980

TITLE Calculate Elapsed or Remaining Central Processor (CP) Time.

ENTRY R=XTIME(N)

N integer input control variable;
if N=1, calculate elapsed CP time (in seconds),
if N=2, calculate remaining CP time (in seconds).

R real output variable in which the elapsed or remaining CP time is stored.

EXIT Control is returned to the calling routine storing the elapsed or remaining CP time (in seconds) in R.

XTIME calculates elapsed CP time for the machine on which the job is running (i.e. MFA or MFB) and subtracts this from the job statement limit to calculate remaining time. If the job is running on MFB the CP time parameter on the job statement is multiplied by a factor of 2.5. If an infinite time has been specified (TØ or T32767 on the job statement) XTIME(2) will return 32767. each time XTIME(2) is called.

ACCURACY ±4 ms

SPEED ∿.4 ms (6600, January 1971)

AUTHOR	D.B. Goulding	DATE	January 1971
REVISED	D.G. Stewart	DATE	May 1981

TITLE Calculate Elapsed or Remaining Input/Ouptut (IO) Time.

ENTRY R=TIMEIO(N)

N integer input control variable;
if N=1, calculate elapsed IO time (in seconds),
if N=2, calculate remaining IO time (in seconds).

R real output variable in which the elapsed or remaining IO time is stored.

EXIT Control is returned to the calling routine storing the elapsed or remaining IO time (in seconds) in R.

If an infinite IO time has been specified (IOØ or T32767 on the job statement), TIMEIO(2) will return 32767. each time TIMEIO(2) is called.

ACCURACY ±4 ms

SPEED ∿.4 ms

AUTHOR	D.G. Stewart	DATE	November 1974
REVISED	D.G. Stewart	DATE	May 1981

TITLE Mainframe Identification Subroutine

Since jobs execute on only one mainframe, either the CYBER 170 or the CDC 6600, the mainframe identifier does not change during a job. This routine should be called at most once during a job.

ENTRY

```
CALL MFID(I)
```

I - integer output variable to return mainframe identification.

EXIT The value of I is set:

I = 1 for MFA (CYBER 170 Model 175)
I = 2 for MFB (6600)

and control is returned to the calling routine.

EXAMPLE The initialization sequence

```
  .
  .
  .
CALL MFID(I)
IF(I.EQ.1)TC=TC/2.5
  .
  .
  .
```

modifies 6600-derived timing constant TC if the program is executing on the 175.

AUTHOR N.J. Abush DATE 9 February 1977

TITLE

Return Current Jobname to a User's Program.

INTRODUCTION

This COMPASS routine sets up a call to the system routine LDL which reads word 25 from the Control Point Area. Word 25 contains the current jobname in the left most seven characters (42 bits) with other information in the right three characters (18 bits).

Because this routine uses information stored by the operating system, use it only when absolutely necessary and make sure that its use is adequately documented in your program. Also, it should only be called once in a job if it is required.

Also note that a change to the Control Point Area or the system routine LDL could prevent JOBNAME from executing properly.

ENTRY

I=JOBNAME(J)
or
CALL JOBNAME(J)

where J is an output variable to return the current jobname, left justified blank filled.

ROUTINES CALLED

System routine LDL.

EXIT

The current jobname is returned left justified blank filled in J and as the function value, I. No error checking is done.

PROGRAM EXAMPLE

The sequence

```
      .
      .
      .
      I=JOBNAME(J)
      PRINT 1000,I
 1000 FORMAT(≠JOBNAME IS≠,A7)
      .
      .
      .
```

will print the current jobname in the user's output file.

AUTHOR D.G. Stewart DATE December 1978

TITLE

Return the accounting parameters user number, charge number and password to a user's program. Because this routine uses information stored by the operating system, it should be used only when absolutely necessary and its use should be adequately documented in your program. Also it should only be called once in a job. Note that a system change to the control point area could prevent an absolute version of USERAP from executing properly.

INTRODUCTION

The accounting parameter on the 6600/175 job statement has the form

Bc - u/p

where c is the charge number and u is the user number, and p is the password field. This information is stored by the operating system in the control point (CP) area. USERAP is a utility routine that returns the user number, charge number, and password as obtained from the control point (CP) area not directly accessible from the users program.

ENTRY

CALL USERAP (UN,CN,PW)

where UN is an output variable (real or integer) in which the user number is returned left justified blank filled,
CN is an output variable (real or integer) in which the charge number is returned left justified blank filled,
PW is an output variable (real or integer) in which the password is returned left justified blank filled. If the password field is not used, all blanks will be returned.

ROUTINES CALLED

AEISER	AELIB routine to ussue AELIB error messages from COMPASS routines
ACT	is a PP routine loaded by the system that accesses words in the control point area.

EXIT

NORMAL EXIT: The current user number, charge number and password are returned left justified blank filled in UN, CN, and PW respectively.

ERROR EXIT: If USERAP is called with fewer than 3 arguments the message

```
***ERROR IN USERAP - LESS THAN 3 ARG IN CALL
```

is issued with traceback and only the number of arguments in the call will be defined.

PROGRAM EXAMPLE: The following simple job sequence, with output illustrates the use of USERAP.

```
TUSERAP,B235-02626/AELIB,T5,IO5.
FTN(R=0)
LGO.
      PROGRAM TUSERAP(OUTPUT)
C
      CALL USERAP(UN,CN,PW)
      PRINT 1,CN,UN,PW
  1   FORMAT(1H1,//,17H CHARGE NUMBER = ,A10,/,
     2            15H USER NUMBER = ,A10,/,
     3            12H PASSWORD = ,A10)
      STOP
      END
```

```
CHARGE NUMBER = 0235
USER NUMBER = 02626
PASSWORD = AELIB
```

EXTERNAL VERSION: A commented do-nothing version of TRACEB is available on the AELIB source tape for inclusion in programs being sent to other computing installations. To retrieve this version, include the UPDATE directive *DEFINE EXTERNAL in the job that requests the source code.

Authors: D. McPherson & D.G. Stewart
1982 January

TITLE

Monitor Program Execution to see where a Program is Spending its CP Time.

INTRODUCTION

SPY is a peripheral processor program which can monitor a program executing in the central processor. It uses a sampling technique to gather data on the amount of time spent executing in different sections of the code and writes this data on a disk file called DOSSIER in a form suitable for presentation as a histogram. The subroutines SPYON and SPYOFF control the execution of SPY. The central processor program PRNTSPY is used to analyze the data and print the resulting histogram.

ENTRIES

(1) Turn SPY on.

CALL SPYON (LOW,IHIGH,NAME,IBINW)

LOW — integer variable specifying the address of the start of the area to be monitored.

IHIGH — integer variable specifying the address of the end of the area to be monitored.

NAME — integer variable containing a left justified eight character display code string which will appear in the title of the SPY output.

IBINW — integer variable specifying the width of the bin to be used in sampling. Bin width must be a power of 2 between 1 and 64 and therefore can be any of the following values, 1B,2B,4B,10B,20B,40B,100B.

(2) Turn SPY off.

CALL SPYOFF

Note: Each time SPYOFF is called, the data accumulated by SPY is written to the file DOSSIER as one logical record.

(3) To print the histogram accumulated by SPY, use the control statement:

```
PRNTSPY.
```

The program PRNTSPY reads each record from the histogram file DOSSIER, analyses the data and produces one of two possible reports, depending on whether the executed program was loaded in relocatable or absolute form.

If a relocatable load took place then information about the location and name of each routine is available on a system created file called ZZZZZ17. PRNTSPY reads this file and adds the names of the routines to the basic histogram display so that the user can see easily in which routines the program was spending its time.

If an absolute load took place then a display of the basic histogram is given and the user will have to use the load map produced when the absolute file was created, to relate absolute addresses to the associated routines.

The control statement PRNTSPY (even after an EXIT control statement) does not work if the program terminated with a CP time limit.

SYSTEM RESIDENCE

SPYON, SPYOFF and PRNTSPY reside on the system library NUCLEUS.

EXITS

Normal:

SPY starts accumulating data following a call of SPYON, and terminates when SPYOFF is called, or when the job terminates.

Non-fatal errors:

(1) If a call of SPYOFF is made before a call of SPYON, then the informative diagnostic

```
SPY - TERMINAL CALL BEFORE INITIAL CALL
```

is issued to the dayfile and the job continues.

(2) The maximum number of bins that can be monitored depends on the amount of buffer space available in the peripheral processor running SPY. The present maximum number of bins is 1062_{10} (2046_8). If the number of bins required, i.e. (IHIGH-LOW)/IBINW is greater than the maximum, then SPY uses a value of IHIGH which will allow the maximum number to be used, and prints the following message in the dayfile

```
NUMBER OF BINS REDUCED TO 2046B.
```

Error Exits:

(1) If LOW > IHIGH, the message

```
SPY - PARAMETERS OUT OF ORDER
```

is issued to the dayfile and the job aborts.

(2) If IBINW is not a power of 2, the message

```
SPY - BINWIDTH MUST BE A POWER OF 2
```

is issued to the dayfile and the job aborts.

EXAMPLE

The following job demonstrates the use of "SPY" to monitor where a program is spending its CP time.

```
FTN(R=O)
MAP(OFF)
LGO.
PRNTSPY.
7/8/9
```

```
      PROGRAM TSPY(OUTPUT)
C
      DIMENSION   X(500),Y(500),Z(500)
      COMMON  DUMMY
      DATA  LIMIT/500/
C
C     SET UP THE PARAMETERS FOR THE CALL OF "SPY"
C     USE THE FUNCTION "LOCF" TO DETERMINE FIELD LENGTH
C
      LOW = 0
      IHIGH = LOCF(DUMMY)
      NAME  = 4HTSPY
      IBINW = 40B
C
C     TURN SPY ON
C
      CALL SPYON(LOW,IHIGH,NAME,IBINW)
C
      DO 30 J = 1 , 2
        DO 20 I = 1 , LIMIT
          X(I) = FLOAT(I*J)/100.
   20   CONTINUE
        CALL EXPON(X,Y,LIMIT)
        CALL GETLOG(Y,Z,LIMIT)
   30 CONTINUE
C
C     TURN SPY OFF
C
      CALL SPYOFF
C
      END

      SUBROUTINE GETLOG(Y,Z,LIMIT)
C
      DIMENSION Y(1),Z(1)
C
C     ARRAY B IS PRESENT SOLELY TO INCREASE THE LENGTH OF THIS ROUTINE
C
      DIMENSION B(50)
      DO 10 I = 1 , LIMIT
        Z(I) = ALOG(Y(I))
   10 CONTINUE
      RETURN
      END

      SUBROUTINE EXPON(X,Y,LIMIT)
C
      DIMENSION X(1),Y(1)
C
C     ARRAY A IS PRESENT SOLELY TO INCREASE THE LENGTH OF THIS ROUTINE
C
      DIMENSION A(50)
      DO 10 I = 1 , LIMIT
        Y(I) = EXP(X(I))
   10 CONTINUE
      RETURN
      END
```

```
RESULTS OF SPYING ON TSPY

            BIN          NO OF SAMPLES             ELAPSED CPU TIME =      .021 SECONDS
              RECALL           0
             WAITING          60                   SAMPLES WHILE IN EXECUTION          1203
           P <      0B         0    (  0.00%)
           P \  11700B         3    (   .25%)      T(RCL)/T(TOTAL) =    0.00
                                                         BIN WIDTH =  40B
                                         ADDRESS                    55994 SAMPLES/SECOND
                                         RELATIVE
   BIN      PERCENT                      TO START    ROUTINE   ROUTINE   ROUTINE
 RANKING    OF TIME    BIN FWA  BIN LWA  OF ROUTINE   NAME       FWA       LWA

                                                     TSPY         111 -    5206
    6         6.57     2200 -   2237       2067
                                                     EXPON       5207 -    5312
    3        17.87     5200 -   5237         -7
                                                     GETLOG      5313 -    5416
    4        17.71     5300 -   5337        -13
                                                     PT$TERM     5417 -    5450
                                                     SYS=1ST     5451 -    5535
                                                     STP.END     5536 -    5536
                                                     FCL.C.      5537 -    5564
                                                     Q8.IO.      5565 -    5721
                                                     Q2NTRY=     5722 -    5722
                                                     FCL=ENT     5723 -    5762
                                                     FCL=FDL     5763 -    6022
                                                     FEIFST=     6023 -    6025
                                                     FORSYS=     6026 -    6565
                                                     LIBSTAT     6566 -    6732
    5         8.06     6700 -   6737        112
                                                     ALOG.       6733 -    7015
    2        22.53     6740 -   6777          5
                                                     EXP.        7016 -    7107
    1        26.52     7000 -   7037        -16
                                                     EXP.MSG     7110 -    7125
                                                     SYSAID=     7126 -    7126
                                                     FORUTL=     7127 -    7153
                                                     GETFIT=     7154 -    7242
                                                     SPA=        7243 -    7253
                                                     CMF.ALF     7254 -    7433
    8          .08     7240 -   7277        -14
                                                     CMF.CSF     7434 -    7441
                                                     CMM.FFA     7442 -    7455
                                                     CMF.FRF     7456 -    7513
                                                     CMF.GSS     7514 -    7535
                                                     CMM.KIL     7536 -    7547
                                                     CMM.MEM     7550 -    7556
                                                     CMM.R       7557 -    7763
                                                     CMF.SLF     7764 -   10005
                                                     CPSPY$     10006 -   10114
    7          .42    10040 -  10077         32
                                                     CPU.SYS    10115 -   10154
                                                     CTL$RM     10155 -   10602
                                                     ERR$RM     10603 -   10627
                                                     LIST$RM    10630 -   10716
                                                     RM$SYS=    10717 -   10723
                                                     UCLOAD     10724 -   11217
                                                     FDL.COM    11220 -   11233
                                                     FDL.RES    11234 -   11444
                                                     FDL.MMI    11445 -   11666
                                                     LIB.PRG    11667 -   11677
```

SPYON
SPYOFF
PRNTSPY

AUTHOR	Control Data Corporation (CDC)		
REVISED	G.N. Williams D. Cheung	DATE	1978 February 1981 May

TITLE Fetch CCL Register Value

INTRODUCTION This routine reads the value of one of the CDC CYBER Control Language (CCL) registers with the symbolic names R1, R2, R3, and R1G. Together with the AELIB routine SETCCL, it allows access to these registers from a FORTRAN program as well as from control statements. The CYBER Control Language is described in Chapter 5 of the CDC NOS/BE Version 1 Reference Manual, No. 60493800.

ENTRY IVALUE=IGETCCL(RGNAME)

where

RGNAME is an integer input variable, specifying the left justified character string 2LR1, 2LR2, 2LR3, or 3LR1G to specify register R1, R2, R3, or R1G respectively.

IVALUE is an integer variable reporting the value of the specific register.

ROUTINES CALLED

GETCCL, LBYT, SBYT, LIBSTAT and TRACEB from AELIB.

EXIT The current value of the specified register is returned. The parameter RGNAME is checked for validity. If not valid, the message ***ILLEGAL PARAMETER and traceback information are printed, and the job is terminated.

TIMING The central processor time to execute on the 170 is about 40 µs.

EXAMPLE I=IGETCCL(3LR1G)

stores the current value of CCL register R1G in the integer variable I.

AUTHOR E.A. Okazaki DATE 1979 October

TITLE Set CCL Register Value.

INTRODUCTION This routine sets the value of one of the CDC CYBER Control Language (CCL) registers with the symbolic names R1, R2, R3, and R1G. Together with the AELIB routine IGETCCL, it allows access to these registers from a FORTRAN program as well as from control statements. The CYBER Control Language is described in Chapter 5 of the CDC NOS/BE Version 1 Reference Manual, No. 60493800.

ENTRY CALL SETCCL(RGNAME,IVALUE)

where

RGNAME, integer input variable, specifying the left justified character string 2LR1, 2LR2, 2LR3, or 3LR1G to specify register R1, R2, R3, or R1G respectively.

IVALUE, integer input variable, is the value to be stored in the register. The range of values allowed is -131071 to 131071.

SETCCL has a second entry GETCCL that is used by IGETCCL and is not called directly from a user's program.

ROUTINES CALLED

LBYT, SBYT, LIBSTAT and TRACEB from AELIB.
FGETJCI and FSETJCI, utility routines loaded with SETCCL.

EXIT The value IVALUE is stored in the CCL register specified by the RGNAME parameter. If RGNAME is not valid, or if IVALUE is outside the range allowed, the message ***ILLEGAL PARAMETER and traceback information are printed, and the job is terminated.

TIMING The central processor time to execute on the 170 is about 40 μs.

EXAMPLE (1) CALL SETCCL(2LR3,9)
Register R3 is set to 9.

(2)
```
I=IGETCCL(2LR1)+1
CALL SETCCL(2LR1,I)
```

Register R1 is set to its previous value + 1.

AUTHOR E.A. Okazaki DATE 1979 October

TITLE Error Message Text, AELERTX

This text is a table of AELIB error message numbers and corresponding messages. It is used by AELERR to set up and print error messages with traceback. Although users normally should not need to consult this listing, its use may be required to

(i) prepare an AELIB subroutine for external use (see Part A, Appendix 5)

(ii) look up a particular message number before calling AEFREQ.

Although AELERTX is a text not a subroutine, it may be listed in the same way as AELIB subroutines. (See Part A, Section 5). In the system, this text resides on the system library NUCLEUS.

SYSTEM RESIDENCE

NUCLEUS

AUTHOR L.E. Evans DATE April 1978

TITLE Print AELIB Error Messages with Traceback.

ENTRY CALL AELERR(N,FLAG)

N integer input variable, the number of the error message from AELERTX.

FLAG input Hollerith string,
1HF if error is fatal (i.e. job to be aborted)
2HNF if error is non-fatal

ADDITIONAL ENTRY INFORMATION

(1) The labelled common block AELERCM may be used to insert variable text into a message. The common block is defined as

```
COMMON/AELERCM/IERWD(15)
```

Information to be printed in an error message must be encoded into a text string, must be zero byte terminated (i.e. be followed by 12 binary zeroes), and must be stored in successive words of IERWD starting at IERWD(3). Multiple insertions are allowed but each text string must start in a new word.

Insertions are made from AELERCM whenever AELERR encounters the special character ↑ in the message to be printed.

(2) The following programming conventions are to be followed:

(a) The common block AELERCM, if used, is declared using one of two UPDATE common decks on the AELIB source tape

```
*CA ERRORF (if subroutine is written in FORTRAN)
*CA ERRORC (if subroutine is written in COMPASS)
```

(b) Error message numbers passed to AELERR are variables that are assigned and commented at the beginning of the subroutine.

EXIT

If FLAG is 1HF, the error message is printed with traceback via the following call to SYSTEM:

CALL SYSTEM(52,nH--text of message--)

The FORTRAN system then aborts the job.

If FLAG is 2HNF, the error message is printed with traceback via a similar call to SYSTEM:

CALL SYSTEM(51,nH--text of message--)

Control then returns to the calling routine.

ROUTINES CALLED

SYSTEM from the FORTRAN library.

COMMON BLOCKS USED

AELERCM

EXTERNAL VERSION

A version of AELERR that prints error message numbers (using WRITE statements to output unit 6) is available on the AELIB source tape for inclusion in programs being sent to other computing installations. To retrieve this version along with a copy of the list of error messages include the UPDATE directives *DEFINE EXTERNAL and *C AELERTX in the job that requests the source code.

AUTHOR	C.J. Tanner	DATE	July 1975
DOCUMENT REVISED BY	L.E. Evans	DATE	April 1978 April 1980

TITLE Change the Printing Frequency of an AELIB Error Message from a User's Program.

ENTRY CALL AEFREQ(N,M)

N integer input variable,the error number of the message in AELERTX.

M integer input variable specifying the printing interval. The message number N will be printed the first time and subsequently every Mth time it is requested by AELERR.

EXIT AEFREQ maintains a table of printing intervals for AELIB error messages. Unless changed by AEFREQ, this interval is 1 (i.e. messages are always printed). A call to AEFREQ changes the appropriate table entry and returns control to the calling routine.

A call to AEFREQ also triggers the printing of a summary table at job termination. This table specifies the number of times each error message has been called by AELERR with FLAG = 2HNF.

ROUTINES CALLED

AELERR from AELIB.

COMMON BLOCKS USED

AELERCM

EXAMPLE The following subroutine call in a user's program specifies that the AIKINT error message 1831 is to be printed the first time it is requested and every 5th time thereafter:

```
CALL AEFREQ(1831,5)
```

AUTHOR C.J. Tanner DATE July 1975

DOCUMENTATION
REVISED BY L.E. Evans DATE April 1978

ALEROF
ALERON
ALERDM
ALERPR

TITLE

Routines for user control of AELIB error messages that do not result in job termination. (This message control scheme is being introduced gradually into AELIB. Control of error messages using these routines is possible now for only those AELIB routines listed at the end of this writeup.)

The error control routines are:

ALEROF disables (i.e. stops the printing of) error messages that do not result in job termination.

ALERON enables previously disabled error messages.

ALERDM dumps the results of the user's error control requests onto the OUTPUT file.

ALERPR logical function called by AELIB routines to determine whether or not a particular AELIB error message is to be printed.

ENTRIES

```
CALL ALEROF(IROUT,IER)
CALL ALERON(IROUT,IER)
CALL ALERDM
PRINT=ALERPR(IROUT,IER) where PRINT is a logical variable.
```

IROUT integer input variable, the number of the AELIB routine for which message(s) are to be controlled.

By convention, AELIB routines are assigned a five-digit integer number from their classification number as follows:

A-B-C becomes A0B0C
A-BB-C becomes ABB0C
A-BB-CC becomes ABBCC

If IROUT=0,

- ALEROF prints a diagnostic message and ignores the call.
- ALERON turns on all AELIB error messages.
- ALERPR prints a diagnostic message and returns the value .TRUE.

IER integer input variable, the value of the error flag defined by routine IROUT for the message that is to be controlled. If no error flag value is defined by the routine, use IER=0.

If IER=0, then

- ALEROF and ALERON turn off or on, respectively, all messages for the specified routine.
- ALERPR returns the value .TRUE. unless CALL ALEROF(IROUT,0) has been executed by the user more recently than any CALL ALERON(IROUT,0).

ROUTINES CALLED

ALRDEL, ALRINI, ALRFND and ALRTST are utility routines loaded with any of the user error message handling routines.

COMMON BLOCKS USED

/ALRCM1/NLAST,N,ROUTAB(20)/ALRCM2/ERRTAB(20)

The common blocks ALRCM1 and ALRCM2 contain the working table for user control of the printing of error messages and the pointers "nlast" and "n" for this table.

The table (and pointers) are as follows:

	ROUTAB	ERRTAB
(1)		
(NLAST)		
(N)		

NLAST points to be the last used entry.

N points to the physical end of the table.

Each entry in the table contains two integers.

(1) "ROUTAB" is the integer number of an AELIB routine (if it is 0, then the entry is considered empty); and

(2) "ERRTAB" is the integer number of the error message that is to be controlled (if it is 0, and the corresponding entry in "ROUTAB" is not, then the default for all messages for that routine is not to print).

If a message identified by a flag value "IER" in routine "IROUT" is to be printed, then the tables will contain either no entry for that routine, or both of the entries (IROUT,0) and (IROUT,IER). If, however, only one of (IROUT,0) or (IROUT,IER) appears, then the message will not be printed.

The table has been initially set up with 20 entries; this should be enough for most cases. If, however, there is a need for more entries, the table can be expanded by the user.

To allow for easy expansion, the table has been split into two common blocks. As a result of this layout, there are only two things that have to be done to expand the table:

(1) Define the common blocks in a (user) routine with the arrays "ROUTAB" and "ERRTAB" dimensioned to the required size, and

(2) Set "N" to the new size and "NLAST" to 0. This may be done in a data statement. This should be done before any call to ALEROF, ALERON, or ALERPR.

**Note:

If you are using overlays, you will have to make sure that the common blocks /ALRCM1/ and /ALRCM2/ are defined in the (0,0) overlay.

EXIT

A normal exit returns control to the calling program after modifying or interrogating the error message table. If errors are detected in the arguments or if the table overflows, a message will be printed with traceback. The errors are non-fatal and will result in the call being ignored (except for a call to ALERON which causes the table to overflow; in this case, all messages for the specified routine are enabled). The possible error messages are:

***TOO MANY ERROR CONTROL REQUESTS - EXPAND THE TABLE IF NECESSARY

All table entries have been used (20 unless increased by user) and the latest request could not be accommodated. To remedy this, expand the table as described above. Note that the table, as described above, usually requires one entry for each message that is specifically turned on or off, and one entry for each routine that has, as default, not to print.

***OPTION OF TURNING OFF ALL ERROR MESSAGES NOT SUPPORTED

A call of the form CALL ALEROF(0,x), where x is any number, has been executed.

***INVALID ARGUMENT - ROUTINE NUMBER IS ZERO

A call of the form PRINT=ALERPR(0,x), where x is any number, has been executed. ALERPR will return a value of .TRUE.

EXAMPLES

(1) CALL ALEROF(020311,1)

This call prevents the printing of the ORDERF/ORDERB error message associated with an error flag value of 1,

***ARRAY INDICES I=_, J=_ ARE NEGATIVE, ZERO, OR I.GT.J

(2) CALL ALEROF(011120,0)
CALL ALERON(011120,1)

These calls make sure that the only non-fatal error message that MLSQQ will print is the one associated with an error flag value of 1,

***INVALID OPTION SELECTED - IOPT=0 ASSSUMED.

(3) CALL ALERON(0,0)

This call will turn on all error messages.

AUTHOR P. Bumbulis DATE May 1979

List of AELIB Routines for which this Message Control Can be Used

PSATLW	1-4-00
TSATLW	1-4-01
STENLW	1-4-02
TCONLW	1-4-03
VISLW	1-4-04
CMMULT/CMADD/CMSUB	1-7-02
CIPLOT	1-11-16
MLSQQ	1-11-20
NL2INT	1-11-24
NL2SOL/NL2SNO	1-11-25
RADCURV	1-11-30
SMOOTH	1-12-00
AIKINT	1-14-02
SIMPLT/SIMPLTM	2-1-20
CONTOUR	2-1-35
SCALE	2-1-66
ORDER/ORDERB	2-3-11
DMPFET	2-5-00
OFLOW	2-5-10
RELESE	2-5-20
ROUTE	2-5-21
CHNGFX	2-5-32
CHNGEX	2-5-33
CHNGEC	2-5-36
USERAP	2-5-44
AEISER	2-5-76
SETBAD	2-5-77

TITLE Issue a traceback report following AELIB error messages.

ENTRY CALL TRACEB(IABORT)

IABORT integer input variable which determines where control is to be returned after printing a message and traceback.

=0, return control to the calling routine.
≠0, abort user's job.

ROUTINES CALLED

SYSTEM is a FORTRAN library routine called to issue traceback.

EXITS There are two possible exits from TRACEB. These are:

(1) <u>Non-fatal Exit</u>

If IABORT=0, control is returned to the user's program after the message

THIS AELIB ERROR, REPORTED AS TRACEB ERROR NUMBER 51, ALLOWS THE JOB TO CONTINUE

has been printed, and a traceback report issued.

(2) <u>Fatal Exit</u>

If IABORT≠0, TRACEB prints the message

THIS AELIB ERROR, REPORTED AS TRACEB ERROR NUMBER 52, FORCES JOB TERMINATION

with traceback and then aborts the job with a fatal FTN error number 52 that is also reported in the dayfile.

EXAMPLE

Following the printing of an AELIB error message, the call

CALL TRACEB(0)

prints the additional message

```
THIS AELIB ERROR, REPORTED AS TRACEB ERROR NUMBER 51,
ALLOWS THE JOB TO CONTINUE
```

followed by a traceback report and then returns control to the calling program.

EXTERNAL VERSION

A commented do-nothing version of TRACEB is available on the AELIB source tape for inclusion in programs being sent to other computing installations. To retrieve this version, include the UPDATE directive *DEFINE EXTERNAL in the job that requests the source code.

AUTHOR	L.E. Evans	DATE	June 1979
REVISED	L.E. Evans		April 1980

TITLE Monitor the Usage of AELIB Subroutines.

ENTRY

```
CALL LIBSTAT
CALL LIBSTAT
```

(Two calls of LIBSTAT are required to ensure that two consecutive central memory words are reserved for LIBSTAT.)

Note: To retrieve source code for AELIB routines without calls to LIBSTAT to do AELIB usage monitoring, include the UPDATE directive *DEFINE EXTERNAL in the job that retrieves the source code.

ADDITIONAL ENTRY INFORMATION

The following programming conventions are used when calling LIBSTAT:

(1) The calls to LIBSTAT are not coded directly but are called in by one of two UPDATE common decks on the AELIB source tape

```
*CA STATF (if subroutine is coded in FORTRAN)
*CA STATC (if subroutine is coded in COMPASS)
```

(2) The calls to LIBSTAT are the first executable statements following each entry point.

EXIT

LIBSTAT maintains a table of execution counts for up to 20 AELIB routines. When first called from an AELIB routine, LIBSTAT sets up a table entry for that routine, initializes the execution count to 1 and then modifies the call to itself to directly access this table entry. (This is why two successive locations are required.) Subsequent entry to the same AELIB routine will simply increment the appropriate execution count.

On job termination, the FORTRAN system prints special system dayfile messages containing the execution counts. These messages are recovered and analysed to produce statistics on the use of AELIB routines.

LIBSTAT

SYSTEM RESIDENCE	FORTRAN Library		
AUTHOR	D. McPherson	DATE	July 1975
DOCUMENTATION REVISED BY	L.E. Evans	DATE	April 1978 and April 1980

TITLE Issue an Error Message for a COMPASS Program.

INTRODUCTION This routine is provided to allow AELIB COMPASS programs to issue error messages in the standard AELIB fashion and to allow user control over these messages using routines ALEROF and ALERON. (See Section 2.5 Introduction for a Discussion of Error Message Processing Using ALEROF and ALERON.)

The common block AEERCM is provided so that, where a COMPASS subroutine is called by more than one AELIB routine, the routine and message number can be set by the calling routine before it calls the subroutine.

ENTRY To call AEISER from a COMPASS program, both a common block and the COMPASS entry are required:

(a) Common Block:

Use either COMMON/AEERCM/IROUTIN,MESSAG or

```
           USE/AEERCM/
IROUTIN    BSS 1
MESSAG     BSS 1
           USE *
```

where

IROUTIN is the number of the calling AELIB routine. This is derived from the classification number as follows:

A-B-C becomes A0B0C
A-BB-C becomes ABBCC
A-BB-CC becomes ABBCC

MESSAG is message number. If error flags are set by the AELIB routine calling AEISER, then MESSAG should be set to the error flag value for the message to be printed. If error flags are not used, MESSAG should be set to zero.

(b) Calling Sequence:

```
     SA1  ARGX      ARGUMENT LIST
+    RJ   =XAEISER  SUBROUTINE CALL
-    VFD  30/TRACE  FOR TRACEBACK

ARGX CON  FORMAT
     CON  VALUE
     CON  PRFLAG
     DATA 0
```

where

- TRACE is required by FORTRAN traceback. First statement in COMPASS program must be of form

```
TRACE VFD 42/7HPROGRAM,18/ENTRY
```

PROGRAM = program name
ENTRY = address of entry point

- FORMAT is format statement containing error message; e.g.

```
FORMAT DATA H+(≠***THIS IS AN ERROR≠)+
```

- VALUE is optional, a variable to be inserted in the error message if there is one. (PRFLAG specifies whether VALUE is to be specified.)

- PRFLAG=0, do not use VALUE,
  PRFLAG=1, VALUE is to be used.

EXIT

Unless printing of this message has been suppressed by the user calling ALEROF, the message is printed with traceback and control is returned to the calling routine.

ROUTINES CALLED

ALERPR, TRACEB from AELIB.

COMMON BLOCKS USED

AEERCM contains routine and message (i.e. error flag) numbers from calling routine as defined above.

EXAMPLE Users who wish to use AEISER in their own programs should discuss their application with their site consultant. Specialized examples of use can be made available if necessary.

AUTHOR C.J. Tanner DATE 1979 September

TITLE

Set Variables or Array Elements to Indefinite Following an AELIB Error in Which No Best Output Values can be Defined. (The indefinite pattern 600 ... 0_8 is used to force a mode 4 error if these output values are used in subsequent arithmetic calculations.)

ENTRY

CALL SETBAD(A,NR,LR,LC)

A real output variable or array returned by SETBAD set to indefinite.

NR integer input variable. Number of rows in array A as dimensioned in the calling routine if A is a two-dimensional array. If A is a variable or a singly dimensioned array, then use NR=1.

LR integer input variable. Number of rows in A to be set indefinite if A is a two-dimensional array. If A is a singly dimensioned array, LR should be the number of elements to be set indefinite. If A is a variable, use LR=1.

LC integer input variable. Number of columns in A to be set indefinite if A is a two-dimensional array. If A is a variable or a singly-dimensioned array, use LC=1.

SETBAD is principally used by AELIB routines to set output parameters of these routines to indefinite in the event of errors for which output parameters have no best value. Should SETBAD be called by a user's routine, care should be taken not to over index the input array and to double the size of NR if A is declared COMPLEX or DOUBLE PRECISION in the calling routine. Failure to heed this advice will lead to unexpected results.

ROUTINES CALLED

ALERPR, TRACEB and LIBSTAT from AELIB.

EXIT

A normal exit returns control to the calling routine with A set indefinite. If NR, LR or LC < 1, the following error message accompanied by the current values of NR, LR and LC is printed complete with traceback.

```
***INPUT PARAMETER ERROR.  NR<1, LR<1 OR LC<1 IS INVALID.
NR=    LR=    LC=
```

Control is then returned to the calling routine without setting values in A.

EXAMPLE

(1) This example illustrates how to call SETBAD to set the first five rows and columns of the two-dimensional array A to indefinite.

```
REAL A(10,10)
INTEGER NA,ROWSA,COLSA
NA=10
ROWSA=COLSA=5
CALL SETBAD(A,NA,ROWSA,COLSA)
```

(2) This example illustrates how to call SETBAD to set the first ten entries in the single-dimensioned array B to indefinite.

```
B(20)
INTEGER NB,ROWSB,COLSB
NB=COLSB=1
ROWSB=10
CALL SETBAD(A,NB,ROWSB,COLSB)
```

EXTERNAL VERSION

A commented do-nothing version of SETBAD is available on the AELIB source tape for inclusion in programs being sent to other computing installations. To retrieve this version, include the UPDATE directive *DEFINE EXTERNAL in the job that requests the source code.

AUTHOR G.L. Klawitter and E.G. Long DATE January 1980

COMMENT SHEET

Title: AELIB Users' Manual

Publication No: AECL-6076 Revision: D

Your comments on this report are welcome. Any errors, suggestions, and comments may be noted below. Please send to

Mathematics & Computation Branch
Stn. 79
Atomic Energy of Canada Limited, Research Company
Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0
Canada

From

Name: ______________________________

Address: (if CRNL, station number is sufficient) ______________

____

____

Comments