

March 1975

NCAR Software Support Library Volume 1

Editors: Jeanne C. Adams
Alan K. Cline
Margaret A. Drake
Roland A. Sweet

ATMOSPHERIC TECHNOLOGY DIVISION
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
BOULDER, COLORADO

(1) NONLINEAR SYSTEMS/POLYNOMIAL ROOTFINDERS

(2) INTERPOLATION/APPROXIMATION/SMOOTHING

(3) LINEAR ALGEBRA

(4) QUADRATURE

FOREWORD

The *NCAR Scientific Subroutine Library* (NSSL) is a collection of routines available to users from the system file library called ULIB. The mathematical routines include many of the algorithms frequently used in scientific computation. There are utility routines and special-purpose routines to facilitate program input/output. The graphics routines provide easy access to a variety of on/line graphical techniques including contouring. This manual contains documentation for users and is divided by topic into thirteen chapters included in three volumes. The following table shows the chapters that are included in each volume.

- In Volume 1:
1. Solution of Non-Linear Systems.
Determination of Roots of Polynomials.
 2. Interpolation, Approximation and Smoothing.
 3. Solution of Linear Systems and Eigenvalue/
Eigenvector Analysis.
 4. Numerical Integration (Quadrature).
- In Volume 2:
5. Solution of Ordinary Differential Equations.
 6. Solution of Partial Differential Equations.
 7. Evaluation of Special Mathematical Functions.
 8. Fast Fourier Analysis.
 9. Statistical Analysis and Random Number Generation.
- In Volume 3:
10. Special-Purpose Input/Output Routines.
 11. Data Processing Utility Routines.
 12. Computer Graphics.
 13. File Manipulation, Text Editing, Program Preprocessing,
and Debugging.

Each program write-up describes how to gain access to a particular routine and how to use it. There are descriptions of the algorithm used in the routine or a reference to the literature on the algorithm. Comment cards at the beginning of each routine repeat some of the material in the manual write-up. These cards always contain the parameters used and their functions. Some chapters, like Graphics, are preceded by an introduction which orients the user to techniques that may be unfamiliar and introduces the facilities available in the routine, pointing out the options, differences and similarities among the routines.

Introductory material for each volume includes the Table of Contents and Description of Subroutines for all three volumes. Those chapters which are included in a given volume will be printed in bold face type (**CHAPTER 1**) while those chapters not included will be in small type and enclosed in brackets ([CHAPTER 5]). Indexing keys to chapters are placed only at the edge of chapters in the Table of Contents that are included in a given volume.

The NSSL routines are stored on ULIB as source cards in COSY (blanks suppressed) form. The job control language provides access to the routine, the access cards are listed in the documentation. Any routine on ULIB may be read or changed by editing. However, users cannot permanently change the contents of any file since these files are READ ONLY. Errors or suggestions for changes to the files should be reported to the librarian or consultant on duty in the Library and Consultation Office.

The NSSL Routines are complemented by the *International Mathematical and Statistical Library* (IMSL) which is rented by NCAR. The IMSL library is on PLIB. Techniques or algorithms not found in NSSL may be available in the IMSL library. Complete documentation (which is available to users in the Library and Consultation Office) can be ordered from IMSL (see *A User's Guide to the NCAR Computing Facility*, Chapter 12). NCAR contracts for the IMSL with the provision that it be used at NCAR. No user should attempt to transport this library to a facility outside of NCAR.

The *NCAR Scientific Subroutine Library* has been created, documented, standardized and checked by the Programming Staff at NCAR. This manual has taken a year to prepare, but many of the routines it contains have been in preparation, improved, used and modified for several years and we expect to add new routines as they become available. When a routine is added to this library, it will be supported by the staff, that is, corrected or changed and maintained.

Assistance in the use of these routines can be obtained in the Library and Consultation Office. Users should also report suspected errors or improvements in already documented routines and suggestions for new routines to the Consultation Staff.

LIST OF SUBROUTINE SPECIALISTS

Jeanne C. Adams

LCKSUM (11)

John Adams

DBNDZRO (1)
DCPOLY (1)
DRPOLY (1)

Dan Anderson

AUTOGRAPH (12)

Jay W. Chalmers

MOVE (11)
PWRX (12)
VELVEC (12)

Bang Yaw Chin

EIGRFA (3)
EIGSFM (3)
EIGSTM (3)
EIGSTS (3)

Pat Coyle

SCROLL (12)

Astrik Deirmendjian

RTNI (1)

Margaret Drake

ELIPE (7)
ELIPK (7)
EXPINT (7)

Dean Frey

TAPECY (10)

William B. Frye

SIMPSN (4)

Dave Fulker

FFT (8)
FFTPOW2 (8)
SPLPAK (2)

Bonnie Gacnik

CONRECQCK (12)
CONRECSMTH (12)
DASHSMTH (12)

Gilbert Green

BRANRD (10)

Jordan Towner Hastings

BSL1NT (2)
BSL2NT (2)
HRM1NT (2)
HRM2NT (2)
READLX (10)

Roy Jenne

DATE (11)
HOURS (11)
UZZBLOK (10)

Dennis Joseph

CHCONV (11)
CONV360 (11)

David Kennison

EDITOR (13)
 FLEX (13)
 FRED (13)
 UCHAR (11)

Robert Lackman

SPAL (9)
 SPECFT (9)

Jack Miller

AML (5)
 RKL (5)
 RNDEV (9)

Ned Read

BESLIK (7)
 BESLJY (7)
 CXERFC (7)
 HYPER (7)

Russell K. Rew

CURV (2)
 FIDEL (13)
 GAUSS (4)
 KURV (2)
 KURVP (2)
 QURV (2)
 SURF (2)

E. Cicely Ridley

ADQUAD (4)
 CUBSPL (2)
 DIFSUB (5)
 GEAR (5)
 SUPMAP (12)
 SUPRLS (3)
 TRIANGLE (2)

Richard K. Sato

LINEQSV (3)

Paul Swarztrauber

BLKTRI (6)
 PWSCSP (6)
 PWSSSP (6)

Roland Sweet

POIS (6)
 PWSCRT (6)
 PWSCYL (6)

Jo Walsh

BSEARCH (11)
 CHLSLV (3)
 CNFPRB (9)
 HSHSLV (3)
 INVMTX (3)
 RGRSN1 (9)
 RGRSN2 (9)
 RGRSN3 (9)
 RGRSN4 (9)
 RGRSN5 (9)
 RGRSN6 (9)
 RLHPTS (9)
 SVDSLX (3)

Nancy Werner

BDSLX (3)
 BND3 (3)
 EIGCFA (3)
 EIGHFS (3)
 EIGSFS (3)
 TRDI (3)
 TRDIP (3)

Marie Working

CORFOR (10)
 UBLOK (10)

Thomas Wright

CONREC Standard (12)
CONRECQCK (12)
CONRECSMTH (12)
DASHCHAR (12)
DASHLINE (12)
HAFTON (12)
ISOSRF (12)
ISOSRFHR (12)
PWRX (12)
PWRY (12)
PWRZ (12)
SRFACE (12)

Numbers in parentheses indicate chapter number in NSSL manual.

CONTENTS
VOLUME 1

CHAPTER 1	SOLUTION OF NON-LINEAR SYSTEMS DETERMINATION OF ROOTS OF POLYNOMIALS	
	DBDNZRO	1.DBDNZRO.1
	DCPOLY	1.DCPOLY.1
	DRPOLY	1.DRPOLY.1
	RTNI	1.RTNI.1
CHAPTER 2	INTERPOLATION, APPROXIMATION AND SMOOTHING	
	BSL1NT	2.BSL1NT.1
	BSL2NT	2.BSL2NT.1
	CUBSPL	2.CUBSPL.1
	CURV	2.CURV.1
	HRM1NT	2.HRM1NT.1
	HRM2NT	2.HRM2NT.1
	KURV	2.KURV.1
	KURVP	2.KURVP.1
	QURV	2.QURV.1
	SPLPAK	2.SPLPAK.1
	SURF	2.SURF.1
	TRIANGLE	2.TRIANGLE.1
CHAPTER 3	SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/ EIGENVECTOR ANALYSIS	
	BDSLV	3.BDSLV.1
	BND3	3.BND3.1
	CHLSLV	3.CHLSLV.1
	EIGCFA	3.EIGCFA.1
	EIGHFS	3.EIGHFS.1
	EIGRFA	3.EIGRFA.1
	EIGSFM	3.EIGSFM.1
	EIGSFS	3.EIGSFS.1
	EIGSTM	3.EIGSTM.1
	EIGSTS	3.EIGSTS.1
	HSHSLV	3.HSHSLV.1
	INVMIX	3.INVMIX.1
	LINEQSV	3.LINEQSV.1
	SUPRLS	3.SUPRLS.1
	SVDSLV	3.SVDSLV.1
	TRDI	3.TRDI.1
	TRDIP	3.TRDIP.1

CHAPTER 4	NUMERICAL INTEGRATION (QUADRATURE)	
	ADQUAD	4.ADQUAD.1
	GAUSS	4.GAUSS.1
	SIMPSN	4.SIMPSN.1
[CHAPTER 5]	SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS	
	AML	5.AML.1
	DIFSUB	5.DIFSUB.1
	GEAR	5.GEAR.1
	RK1	5.RK1.1
[CHAPTER 6]	SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS	
	BLKTRI	6.BLKTRI.1
	POIS	6.POIS.1
	PWSCRT	6.TWSCRT.1
	PWSCSP	6.PWSCSP.1
	PWSCYL	6.PWSCYL.1
	PWSSSP	6.PWSSSP.1
[CHAPTER 7]	EVALUATION OF SPECIAL MATHEMATICAL FUNCTIONS	
	BESLIK	7.BESLIK.1
	BESLJY	7.BESLJY.1
	CXERFC	7.CXERFC.1
	ELIPE	7.ELIPE.1
	ELIPK	7.ELIPK.1
	EXPINT	7.EXPINT.1
	HYPER	7.HYPER.1
[CHAPTER 8]	FAST FOURIER ANALYSIS	
	FFT	8.FFT.1
	FFTPOW2	8.FFTPOW2.1
[CHAPTER 9]	STATISTICAL ANALYSIS AND RANDOM NUMBER GENERATION	
	CNFPRB	9.CNFPRB.1
	RGRSN1	9.RGRSN1.1
	RGRSN2	9.RGRSN2.1
	RGRSN3	9.RGRSN3.1
	RGRSN4	9.RGRSN4.1
	RGRSN5	9.RGRSN5.1
	RGRSN6	9.RGRSN6.1
	RLHPTS	9.RLHPTS.1
	RNDEV	9.RNDEV.1
	SPAL	9.SPAL.1
	SPECFT	9.SPECFT.1

[CHAPTER 10]	SPECIAL-PURPOSE INPUT/OUTPUT ROUTINES	
	BRANRD	10.BRANRD.1
	CORFOR	10.CORFOR.1
	READLX	10.READLX.1
	TAPECY	10.RAPECY.1
	UBLOK	10.UBLOK.1
	UZBLOK	10.UZBLOK.1
[CHAPTER 11]	DATA PROCESSING UTILITY ROUTINES	
	BSEARCH	11.BSEARCH.1
	CHCONV	11.CHCONV.1
	CONV360	11.CONV360.1
	DATE	11.DATE.1
	HOURS	11.HOURS.1
	LCKSUM	11.LCKSUM.1
	MOVE	11.MOVE.1
	UCHAR	11.UCHAR.1
[CHAPTER 12]	COMPUTER GRAPHICS	
	INTRODUCTION	
	AUTOGRAPH	12.AUTOGRAPH.1
	CONREC Standard	12.CONREC STANDARD.1
	CONRECQCK	12.CONRECQCK.1
	CONRECSMTH	12.CONRECSMTH.1
	DASHCHAR	12.DASHCHAR.1
	DASHLINE	12.DASHLINE.1
	DASHSMTH	12.DASHSMTH.1
	HAFTON	12.HAFTON.1
	ISOSRF	12.ISOSRF.1
	ISOSRFHR	12.ISOSRFHR.1
	PWRX	12.PWRX.1
	PWRY	12.PWRY.1
	PWRZ	12.PWRZ.1
	SCROLL	12.SCROLL.1
	SRFACE	12.SRFACE.1
	SUPMAP	12.SUPMAP.1
	VELVEC	12.VELVEC.1
	APPENDIX 1: PROCESSING ARRAYS	
	APPENDIX 2: TRANSFORMATIONS	
[CHAPTER 13]	FILE MANIPULATION, TEXT EDITING, PROGRAM PREPROCESSING AND DEBUGGING	
	EDITOR	13.EDITOR.1
	FIDEL	13.FIDEL.1
	FLEX	13.FLEX.1
	FRED	13.FRED.1

DESCRIPTION OF SUBROUTINES
VOLUME 1

CHAPTER 1**SOLUTION OF NON-LINEAR SYSTEMS
DETERMINATION OF ROOTS OF POLYNOMIALS**

- DBNDZRO** Computes a posteriori error bounds and improvements to roots of polynomials.
- DCPOLY** Finds all the roots of a polynomial with complex coefficients using double precision.
- DRPOLY** Finds all the roots of a polynomial with real coefficients using double precision.
- RTNI** Finds a root of a given non-linear equation.

CHAPTER 2**INTERPOLATION, APPROXIMATION AND SMOOTHING**

- BSL1NT** Performs one-dimensional Bessel interpolation of selected order for values and derivatives.
- BSL2NT** Performs two-dimensional Bessel interpolation of selected order for values and derivatives.
- CUBSPL** Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives.
- CURV** Performs one-dimensional interpolation, differentiation, and integration using splines under tension.

CHAPTER 2
(continued)

INTERPOLATION, APPROXIMATION AND SMOOTHING (continued)

HRMINT	Performs one-dimensional Hermite interpolation of selected order for values and derivatives.
HRM2NT	Performs two-dimensional Hermite interpolation of selected order for values and derivatives.
KURV	Performs interpolation of a parameterized curve in the plane using splines under tension.
KURVP	Performs interpolation of a parameterized closed curve in the plane using splines under tension.
QURV	Performs interpolation of a parameterized curve in space using splines under tension.
SPLPAK	Performs least squares fitting of multidimensional cubic splines to arbitrarily located data.
SURF	Performs two-dimensional interpolation using a bi-spline under tension.
TRIANGLE	Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane.

CHAPTER 3

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS

BDSL	Solves a banded linear system.
BND3	Solves a tridiagonal linear system using Gaussian elimination with partial pivoting.
CHLSLV	Solves a real, symmetric, positive definite linear system by Cholesky decomposition.

CHAPTER 3
*(continued)***SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS**
(continued)

EIGCFA	Computes all the eigenvalues and selected eigenvectors of a general complex matrix.
EIGHFS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix.
EIGRFA	Computes all the eigenvalues and selected eigenvectors of a general real matrix.
EIGSFM	Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric matrix.
EIGSFS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix.
EIGSTM	Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix.
EIGSTS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix.
HSISLV	Solves a real overdetermined linear system using Householder transformations.
INVTX	Computes the inverse of a general real matrix using Gaussian elimination with full pivoting.
LINEQSV	Solves a real linear system using Gaussian elimination with partial pivoting.

CHAPTER 3
(continued)

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS
(continued)

- SUPRLS** Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.
- SVDSL** Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.
- TRDI** Solves a diagonally dominant tridiagonal linear system using Gaussian elimination without pivoting.
- TRDIP** Solves a diagonally dominant periodic tridiagonal linear system using Gaussian elimination without pivoting.

CHAPTER 4

NUMERICAL INTEGRATION (QUADRATURE)

- ADQUAD** Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of functional evaluations required.
- GAUSS** Calculates Gaussian quadrature abscissae and weights relative to a given weight function on a given finite or infinite interval of integration.
- SIMPSN** Integrates a function using Simpson's rule if given at equally spaced abscissae or using interpolatory quadratics if given at unequally spaced abscissae.

SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

- AMI** Performs one step of the implicit fourth order Adams-Moulton method. No error control.
- DIFSUB** Performs one variable step using various methods. Good for stiff systems. Good error control
- GEAR** Performs one step using Gear's subroutine, DIFSUB. Built-in error control.
- RKI** Performs one step of the explicit fourth order Runge-Kutta-Gill method. No error control.

[CHAPTER 6]

SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

- BLKTRI** Solves a separable elliptic equation.
- POIS** Solves an elliptic equation with variable coefficients on one derivative.
- PWSCRT** Solves two-dimensional Helmholtz equation in Cartesian coordinates.
- PWSCSP** Solves a two-dimensional Helmholtz equation in spherical coordinates assuming no functional dependence on longitude.
- PWSCYL** Solves two-dimensional Helmholtz equation in cylindrical coordinates.
- PWSSSP** Solves two-dimensional Helmholtz equation on the surface of a sphere.

[CHAPTER 7]

EVALUATION OF SPECIAL MATHEMATICAL FUNCTIONS

- BESLIK** Computes modified Bessel functions of the first and second kind for real argument and real order.
- BESLJY** Computes Bessel functions of the first and second kind for real argument and real order.
- CXERFC** Computes the complex complementary error function of a complex argument.
- ELIPE** Computes complete elliptic integrals of the second kind.
- ELIPK** Computes complete elliptic integrals of the first kind.
- EXPINT** Computes exponential integrals.
- HYPER** Computes hyperbolic functions.

[CHAPTER 8]

FAST FOURIER ANALYSIS

- FFT** Computes fast Fourier transforms for data vectors of arbitrary length.
- FFTPOW2** Computes fast Fourier transforms for data vectors whose length is a power of two.

STATISTICAL ANALYSIS AND RANDOM NUMBER GENERATION

- CNFPRB** Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients.
- RGRSN1** Calculates a set of regression coefficients for an unweighted regression model.
- RGRSN2** Weights the model with a diagonal matrix of weights.
- RGRSN3** Weights the model with the covariance matrix of the random variables.
- RGRSN4** Similar to RGRSN3 except that the covariance matrix is banded.
- RGRSN5** Weights the model using standard deviations of repeated observations.
- RGRSN6** Weights the model using an estimated covariance matrix from repeated observations.
- RLHPTS** Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable.
- RNDEV** Generates independent random deviates with mean 0 and variance 1.
- SPAL** Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data.
- SPECFT** Calculates a single power spectrum of an input time series or the cross spectrum of two input time series, including cospectrum, quadrature, coherence, and phase.

[CHAPTER 10]

SPECIAL-PURPOSE INPUT/OUTPUT ROUTINES

BRANRD	Provides buffered random access I/O.
CORFOR	Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted.
READLX	Provides free format data-directed input and program control facilities.
TAPECY	Copies, combines, and edits tapes.
UBLOK	Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries.
UZBLOK	Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries.

[CHAPTER 11]

DATA PROCESSING UTILITY ROUTINES

BSEARCH	Performs binary search of a table of floating point numbers.
CHCONV	Converts characters in one character set to another character set by indexing a conversion table.
CONV360	Converts IBM 360 (EBCDIC) cards to CDC (BCD) cards.
DATE	Computes date information from number of hours after December 31, 1920.
HOURS	Computes hours since December 31, 1920, from date information.

[CHAPTER 11]
(continued)

DATA PROCESSING UTILITY ROUTINES (continued)

- LCKSUM** Provides a fast checksum of data.
- MOVE** Provides a rapid in-core transfer of data.
- UCHAR** Provides a rapid character retrieval facility.

[CHAPTER 12]

COMPUTER GRAPHICS

- AUTOGRAPH** Draws and annotates curves or families of curves.
- CONREC Standard** Contours two-dimensional arrays, labelling the contour lines.
- CONRECQCK** Like CONREC Standard, but faster and smaller because contours are unlabelled.
- CONRECSMTH** Like CONREC Standard, but bigger and slower because contours are smoothed as well as labelled.
- DASHCHAR** Software dashed line package with labelling capability.
- DASHLINE** Like DASHCHAR, but smaller and faster because it has no labelling capacity.
- DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed.
- HAFTON** Halftone (gray scale) pictures from a two-dimensional array.
- ISOSRF** Iso-valued surfaces (with hidden lines removed) from a three-dimensional array.

[CHAPTER 12]
(continued)

COMPUTER GRAPHICS (continued)

ISOSRFHR	Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array.
PWRX	High quality software characters.
PWRY	Simplest software characters.
PWRZ	Three-space characters for use with ISOSRF or SRFACE.
SCROLL	Movie titling package.
SRFACE	Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array.
SUPMAP	Continental outlines and political boundaries in various projections.
VELVEC	Two-dimensional velocity field displayed by drawing arrows from the data locations.

[CHAPTER 13]

FILE MANIPULATION, TEXT EDITING, PROGRAM PREPROCESSING AND DEBUGGING

EDITOR	Program to maintain a tape library of source card files in a PLIB-compatible form.
FIDEL	Pre-compiler for the language PDELAN (an extension of FORTRAN to facilitate implementation of finite difference approximations to partial differential equations).
FLEX	File management of COSYed PLIB card files.
FRED	Pre-compiler for FORTRAN programs providing conditional compilation, macros, debugging aids, evaluation of subscript expressions and renumbering of statement labels.

**ALPHABETICAL LISTING
OF SUBROUTINES****A****ADQUAD**

Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of functional evaluations required. (Volume I, Chapter 4)

AMI

Performs one step of the implicit fourth order Adams-Moulton method. No error control. (Volume II, Chapter 5)

AUTOGRAPH

Draws and annotates curves or families of curves. (Volume III, Chapter 12)

B**BDSLV**

Solves a banded linear system. (Volume I, Chapter 3)

BESLIK

Computes modified Bessel functions of the first and second kind for real argument and real order. (Volume II, Chapter 7)

BESLJY

Computes Bessel functions of the first and second kind for real argument and real order. (Volume II, Chapter 7)

BLKTRI

Solves a separable elliptic equation. (Volume II, Chapter 6)

BND3

Solves a tridiagonal linear system using Gaussian elimination with partial pivoting. (Volume I, Chapter 3)

BRANRD

Provides buffered random access I/O. (Volume III, Chapter 10)

BSEARCH

Performs binary search of a table of floating point numbers. (Volume III, Chapter 11)

BSL1NT

Performs one-dimensional Bessel interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

BSL2NT

Performs two-dimensional Bessel interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

C

- CHCONV** Converts characters in one character set to another character set by indexing a conversion table. (Volume III, Chapter 11)
- CHLSLV** Solves a real, symmetric, positive definite linear system by Cholesky decomposition. (Volume I, Chapter 3)
- CNFPRB** Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients. (Volume II, Chapter 9)
- CONV360** Converts IBM 360 (EBCDIC) cards to CDC (BCD) cards. (Volume III, Chapter 11)
- CONREC Standard** Contours two-dimensional arrays, labelling the contour lines. (Volume III, Chapter 12)
- CONRECQCK** Like CONREC Standard, but faster and smaller because contours are unlabelled. (Volume III, Chapter 12)
- CONRECSMTH** Like CONREC Standard, but bigger and slower because contours are smoothed as well as labelled. (Volume III, Chapter 12)
- CORFOR** Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted. (Volume III, Chapter 10)
- CUBSPL** Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives. (Volume I, Chapter 2)
- CURV** Performs one-dimensional interpolation, differentiation, and integration using splines under tension. (Volume I, Chapter 2)
- CXERFC** Computes the complex complementary error function of a complex argument. (Volume II, Chapter 7)

D

- DASHCHAR** Software dashed line package with labelling capability. (Volume III, Chapter 12)
- DASHLINE** Like DASHCHAR, but smaller and faster because it has no labelling capacity. (Volume III, Chapter 12)
- DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed. (Volume III, Chapter 12)
- DATE** Computes date information from number of hours after December 31, 1920. (Volume III, Chapter 11)

D (*continued*)

- DBNDZRO** Computes a posteriori error bounds and improvements to roots of polynomials. (Volume I, Chapter 1)
- DCPOLY** Finds all the roots of a polynomial with complex coefficients using double precision. (Volume I, Chapter 1)
- DIFSUB** Performs one variable step using various methods. Good for stiff systems. Good error control. (Volume II, Chapter 5)
- DRPOLY** Finds all the roots of a polynomial with real coefficients using double precision. (Volume I, Chapter 1)
- E**
- EDITOR** Program to maintain a tape library of source card files in a PLIB-compatible form. (Volume III, Chapter 13)
- EIGCFA** Computes all the eigenvalues and selected eigenvectors of a general complex matrix. (Volume I, Chapter 3)
- EIGHFS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix. (Volume I, Chapter 3)
- EIGRFA** Computes all the eigenvalues and selected eigenvectors of a general real matrix. (Volume I, Chapter 3)
- EIGSFM** Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric matrix. (Volume I, Chapter 3)
- EIGSFS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix. (Volume I, Chapter 3)
- EIGSTM** Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix. (Volume I, Chapter 3)
- EIGSTS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix. (Volume I, Chapter 3)
- ELIPE** Computes complete elliptic integrals of the second kind. (Volume II, Chapter 7)
- ELIPK** Computes complete elliptic integrals of the first kind. (Volume II, Chapter 7)

E *(continued)*

EXPINT Computes exponential integrals. (Volume II, Chapter 7)

F

FFT Computes fast Fourier transforms for data vectors of arbitrary length. (Volume II, Chapter 8)

FFTPW2 Computes fast Fourier transforms for data vectors whose length is a power of two. (Volume II, Chapter 8)

FIDEL Pre-compiler for the language PDELAN (an extension of FORTRAN to facilitate implementation of finite difference approximations to partial differential equations). (Volume III, Chapter 13)

FLEX File management of COSYed PLIB card files. (Volume III, Chapter 13)

FRED Pre-compiler for FORTRAN programs providing conditional compilation, macros, debugging aids, evaluation of subscript expressions and renumbering of statement labels. (Volume III, Chapter 13)

G

GAUSS Calculates Gaussian quadrature abscissae and weights relative to a given weight function on a given finite or infinite interval of integration. (Volume I, Chapter 4)

GEAR Performs one step using Gear's subroutine, DIFSUB. Built-in error control. (Volume II, Chapter 5)

H

HAFTON Halftone (gray scale) pictures from a two-dimensional array. (Volume III, Chapter 12)

HOURS Computes hours since December 31, 1920, from date information. (Volume III, Chapter 11)

HRMINT Performs one-dimensional Hermite interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

HRM2NT Performs two-dimensional Hermite interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

HSHSLV Solves a real overdetermined linear system using Householder transformations. (Volume I, Chapter 3)

HYPER Computes hyperbolic functions. (Volume II, Chapter 7)

I

- INVMTX** Computes the inverse of a general real matrix using Gaussian elimination with full pivoting. (Volume I, Chapter 3)
- ISOSRF** Iso-valued surfaces (with hidden lines removed) from a three-dimensional array. (Volume III, Chapter 12)
- ISOSRFHR** Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array. (Volume III, Chapter 12)

K

- KURV** Performs interpolation of a parameterized curve in the plane using splines under tension. (Volume I, Chapter 2)
- KURVP** Performs interpolation of a parameterized closed curve in the plane using splines under tension. (Volume I, Chapter 2)

L

- LCKSUM** Provides a fast checksum of data. (Volume III, Chapter 11)
- LINEQSV** Solves a real linear system using Gaussian elimination with partial pivoting. (Volume I, Chapter 3)

M

- MOVE** Provides a rapid in-core transfer of data. (Volume III, Chapter 11)

P

- POIS** Solves an elliptic equation with variable coefficients on one derivative. (Volume II, Chapter 6)
- PWRX** High quality software characters. (Volume III, Chapter 12)
- PWRY** Simplest software characters. (Volume III, Chapter 12)
- PWRZ** Three-space characters for use with ISOSRF or SRFACE. (Volume III, Chapter 12)
- PWSCRT** Solves two-dimensional Helmholtz equation in Cartesian coordinates. (Volume II, Chapter 6)
- PWSCSP** Solves a two-dimensional Helmholtz equation in spherical coordinates assuming no functional dependence on longitude. (Volume II, Chapter 6)

P *(continued)*

- PWSCYL** Solves two-dimensional Helmholtz equation in cylindrical coordinates. (Volume II, Chapter 6)
- PWSSSP** Solves two-dimensional Helmholtz equation on the surface of a sphere. (Volume II, Chapter 6)

Q

- QURV** Performs interpolation of a parameterized curve in space using splines under tension. (Volume I, Chapter 2)

R

- READLX** Provides free format data-directed input and program control facilities. (Volume III, Chapter 10)
- RGRSN1** Calculates a set of regression coefficients for an unweighted regression model. (Volume II, Chapter 9)
- RGRSN2** Weights the model with a diagonal matrix of weights. (Volume II, Chapter 9)
- RGRSN3** Weights the model with the covariance matrix of the random variables. (Volume II, Chapter 9)
- RGRSN4** Similar to RGRSN3 except that the covariance matrix is banded. (Volume II, Chapter 9)
- RGRSN5** Weights the model using standard deviations of repeated observations. (Volume II, Chapter 9)
- RGRSN6** Weights the model using an estimated covariance matrix from repeated observations. (Volume II, Chapter 9)
- RKI** Performs one step of the explicit fourth order Runge-Kutta-Gill method. No error control. (Volume II, Chapter 5)
- RLHPTS** Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable. (Volume III, Chapter 9)
- RNDEV** Generates independent random deviates with mean 0 and variance 1. (Volume III, Chapter 9)
- RTNI** Finds a root of a given non-linear equation. (Volume I, Chapter 1)

S

- SCROLL** Movie titling package. (Volume III, Chapter 12)
- SIMPSN** Integrates a function using Simpson's rule if given at equally spaced abscissae or using interpolatory quadratics if given at unequally spaced abscissae. (Volume I, Chapter 4)
- SPAL** Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data. (Volume II, Chapter 9)
- SPECFT** Calculates a single power spectrum of an input time series or the cross spectrum of two input time series, including cospectrum, quadrature, coherence, and phase. (Volume II, Chapter 9)
- SPLPAK** Performs least squares fitting of multidimensional cubic splines to arbitrarily located data. (Volume I, Chapter 2)
- SRFACE** Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array. (Volume III, Chapter 12)
- SUPMAP** Continental outlines and political boundaries in various projections. (Volume III, Chapter 12)
- SUPRLS** Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core. (Volume I, Chapter 3)
- SURF** Performs two-dimensional interpolation using a bi-spline under tension. (Volume I, Chapter 2)
- SVDSL** Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core. (Volume I, Chapter 3)

T

- TAPECY** Copies, combines and edits tapes. (Volume III, Chapter 10)
- TRDI** Solves a diagonally dominant tridiagonal linear system using Gaussian elimination without pivoting. (Volume I, Chapter 3)
- TRDIP** Solves a diagonally dominant periodic tridiagonal linear system using Gaussian elimination without pivoting. (Volume I, Chapter 3)

T *(continued)***TRIANGLE**

Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane. (Volume I, Chapter 2)

U**UBLOK**

Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries. (Volume III, Chapter 10)

UCHAR

Provides a rapid character retrieval facility. (Volume III, Chapter 11)

UZBLOK

Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries. (Volume III, Chapter 10)

V**VELVEC**

Two-dimensional velocity field displayed by drawing arrows from the data locations. (Volume III, Chapter 12)

NONLINEAR SYSTEMS/POLYNOMIAL ROOTFINDERS

DBNDZRO

DCPOLY

DRPOLY

RTNI

DBNDZRO

**SUBROUTINE DBNDZRO (NDEG,OPR,OPI,EPS,RLPOLY,TRYIMP,WORK,ZEROR,ZEROI,
NODIST,MULT,FBND,FLAG,IER)**

**DIMENSION OF
ARGUMENTS**

OPR(NDEG+1),OPI(NDEG+1),WORK(52*(NDEG+1)),ZEROR(NDEG),
ZEROI(NDEG),MULT(NDEG),FBND(NDEG),FLAG(NDEG)

LATEST REVISION

April 1, 1974

PURPOSE

DBNDZRO finds individual bounds for a set of approximate
zeros with an improvement option.

ACCESS CARDS

*FORTRAN,S=ULIB,N=DBNDZRO
*COSY

USAGE

CALL DBNDZRO (NDEG,OPR,OPI,EPS,RLPOLY,TRYIMP,WORK,ZEROR,
ZEROI,NODIST,MULT,FBND,FLAG,IER)

ARGUMENTS

On Input

NDEG

The integer degree of the polynomial.

On Input
(continued)

OPR, OPI

Double precision vectors of real and imaginary parts of coefficients. (OPR(1),OPI(1)) is the coefficient of the term of highest degree.

EPS

Double precision maximum relative error in the coefficients. If EPS=0 initially, it is set equal to the machine precision.

RLPOLY

Logical parameter which is TRUE if the polynomial has real coefficients and FALSE if the polynomial has complex coefficients.

TRYIMP

Logical parameter which if TRUE requests that zeros and bounds be improved. FALSE requests that zeros remain unchanged.

WORK

A work space array of dimension 52*(NDEG + 1).

ZEROR, ZEROI

Double precision vectors of initial NDEG approximate zeros. On output the first NODIST locations contain final approximate zeros.

On Output

NODIST

Output number of distinct zeros. This will be NDEG if no improvement is selected.

MULT

Output integer vector where MULT(I) is the multiplicity of the Ith zero.

FBND

Double precision vector of error bounds. The circle centered at (ZEROR(I),ZEROI(I)) contains at least MULT(I) zeros.

FLAG

Logical output vector. If the I^{th} flag is TRUE, the I^{th} circle does not overlap any of the others with FLAG=TRUE. If the I^{th} flag is FALSE, then one or more of the zeros in the I^{th} circle may also be counted in one of the other circles. The union of the TRUE flags contains at least $\sum \text{MULT}(I)$ zeros where the sum is over all values of I with TRUE flags.

IER

An error parameter which is 33 if the leading coefficient is zero and zero otherwise.

ENTRY POINT

DBNDZRO

SPECIAL CONDITIONS

None

COMMON BLOCKS

CONR, CONL, BNDCLI

I/O

ULIBER (on system)

SPECIALIST

John Adams, NCAR, Boulder, Colorado, 80303

LANGUAGE

FORTRAN

HISTORY

Subroutine DBNDZRO was obtained by modifying a polynomial solver package of J. Traub and M. A. Jenkins.

ALGORITHM A discussion of the algorithm "A Three Stage Variable-Shift Iteration for Polynomial Zeros and its Relation to Generalized Rayleigh Iteration" is on file in the Computing Facility Library.

SPACE REQUIRED 10213₈ words

ACCURACY The bounds in array FBND reflect the condition of the zeros. Well conditioned zeros usually produce small error bounds relative to the size of the zeros. Poorly conditioned zeros sometimes produce pessimistic bounds which indicate no significant figures can be guaranteed.

TIMING The time DBNDZRO requires varies considerably depending on the polynomial and whether or not the improvement option is selected.

PORTABILITY The subroutine MICON in DBNDZRO sets the following machine dependent constants

BASE	Base of floating point arithmetic.
PRECIS	Digits in the double precision mantissa.
EXPHI	Maximum exponent.
EXPLO	Minimum exponent.
ROUND	True if double precision arithmetic is rounded and false if chopped.

REQUIRED RESIDENT ROUTINES DSQRT, DCOS, DSIN, DEXP, DLOG

DCPOLY**SUBROUTINE DCPOLY (OPR,OPI,NDEG,WORK,ZEROR,ZEROI,IER)****DIMENSION OF
ARGUMENTS**OPR(NDEG+1),OPI(NDEG+1),ZEROR(NDEG),ZEROI(NDEG),
WORK(28*(NDEG+1))**LATEST REVISION**

January 1, 1974

PURPOSE

DCPOLY calculates the zeroes of a complex polynomial.

ACCESS CARDS*FORTRAN,S=ULIB,N=DCPOLY
*COSY**USAGE**

CALL DCPOLY (OPR,OPI,NDEG,WORK,ZEROR,ZEROI,IER)

ARGUMENTS**On Input**

OPR, OPI

Double precision vectors of real and imaginary parts of
coefficients. (OPR(1),OPI(1)) is the coefficient of the
term of highest degree.

1.DCPOLY.2

On Input
(continued)

NDEG
The integer degree of the polynomial.

WORK
A work space array.

On Output

ZEROR, ZEROI
Double precision vectors of real and imaginary parts of zeroes.

IER
An error parameter where IER
= 0 if all the roots are found.
= 33 if all the roots are not found or the leading coefficient is zero.

ENTRY POINTS DCPOLY

SPECIAL CONDITIONS None

COMMON BLOCKS CONR, RAND, VBLSI, VBLSR

I/O None

PRECISION Double

**REQUIRED ULIB
ROUTINES** ULIBER

SPECIALIST John Adams, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY The subroutines DRPOLY, DCPOLY were obtained by modifying a polynomial solver package conceived and written by J. Traub and M. A. Jenkins.

ALGORITHM A discussion of the algorithm "A Three Stage Variable-Shift Iteration for Polynomial Zeroes and its Relation to Generalized Rayleigh Iteration" is on file in the Computing Facility Library.

SPACE REQUIRED 5407₈ words

ACCURACY A posteriori error bounds for the roots with an option for improvement can be effected using subroutine BNDZRO off ULIB.

TIMING A polynomial of degree N requires approximately $4N^2$ milliseconds on the 7600.

PORTABILITY The subroutine MCON in DCPOLY sets the following machine dependent constants

BASE	Base of floating point arithmetic.
PRECIS	Digits in the double precision mantissa.
EXPHI	Maximum exponent.
EXPLO	Minimum exponent.
ROUND	True if double precision arithmetic is rounded.

PORTABILITY
(continued)

Also, π is set to double precision accuracy in CLPOLY. There are no further restrictions to portability.

**REQUIRED RESIDENT
ROUTINES**

DSQRT, DCOS, DSIN, DEXP, DLOG

DRPOLY**SUBROUTINE DRPOLY (PR,NDEG,WORK,ZEROR,ZEROI,IER)**

DIMENSION OF ARGUMENTS PR(NDEG+1),ZEROR(NDEG),ZEROI(NDEG),WORK(18*(NDEG+1))

LATEST REVISION January 1974

PURPOSE DRPOLY calculates the zeroes of a real polynomial.

ACCESS CARDS *FORTRAN,S=ULIB,N=DRPOLY
*COSY

USAGE CALL DRPOLY (PR,NDEG,WORK,ZEROR,ZEROI,IER)

ARGUMENTS

On Input PR

A double precision vector of real coefficients.
PR(1) is the coefficient of the term of highest degree.

1.DRPOLY.2

On Input

(continued)

NDEG

The integer degree of the polynomial.

WORK

A work space array.

On Output

ZEROR, ZEROI

Double precision vectors of real and imaginary parts of zeroes.

IER

An error parameter where

= 0 If all the roots are found.

= 33 If all the roots have not been found or the leading coefficient is zero.

ENTRY POINTS

DRPOLY

SPECIAL CONDITIONS

None

COMMON BLOCKS

RAND, CONRR, VBLR, VBLL, ARGR, ARGJ

I/O

None

PRECISION

Double

**REQUIRED ULIB
ROUTINES**

ULIBER

SPECIALIST John Adams, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY The subroutines DRPOLY, DCPOLY were obtained by modifying a polynomial solver package conceived and written by J. Traub and M. A. Jenkins.

ALGORITHM A discussion of the algorithm "A Three Stage Variable-Shift Iteration for Polynomial Zeroes and its Relation to Generalized Rayleigh Iteration" is on file in the Computing Facility library at NCAR.

SPACE REQUIRED 4736₈ words

ACCURACY A posteriori error bounds and improvement can be effected using subroutine BNDZRO.

TIMING A polynomial of degree N requires approximately N^2 milliseconds on the 7600.

PORTABILITY The subroutine MCON in DRPOLY sets the following machine dependent constants.

BASE	Base of floating point arithmetic.
PRECIS	Digits in the double precision mantissa.
EXPHI	Maximum exponent
EXPLO	Minimum exponent
ROUND	True if double precision arithmetic is rounded.

1.DRPOLY.4

PORTABILITY
(continued)

Also, π is set to double precision accuracy in RLP1POLY.
There are no further restrictions to portability.

**REQUIRED RESIDENT
ROUTINES**

DSQRT, DCOS, DSIN, DEVP, DLOG

SUBROUTINE RTNI (X,F,DERF,FCT,XST,EPS,IEND,IER)

LATEST REVISION October 1973

PURPOSE To solve general nonlinear equations of the form $F(X)=0$,
by means of Newton's iteration method.

ACCESS CARDS *FORTRAN,S=ULIB,N=RTNI
 *COSY

USAGE CALL RTNI (X,F,DERF,FCT,XST,EPS,IEND,IER)

ARGUMENTS

On Input

XST

Initial guess of the root X.

EPS

Upper bound of the error of the result X.

IEND

Maximum number of iteration steps specified.

FCT

External subroutine provided by the user to calculate the function value F and derivative DERF at given X. Must be declared EXTERNAL in calling program. Its parameter list must be (X,F,DERF).

On Output

X

Resultant root of equation $F(X)=0$.

F

Resultant function value at root X.

DERF

Resultant value of derivative at root X.

IER

Error flag.

= 0 No error.

= 32 No convergence after IEND iteration steps.

= 34 Derivative is zero for some iteration step.

ENTRY POINTS

RTNI

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

ROUTINES TO BE
PROVIDED BY USER

FCT

SUBROUTINE FCT (X,F,DERF)

The subroutine should evaluate

 $F = F(X)$ $DERF = DF(X)/DX$

SPECIALIST Astrik Deirmendjian, NCAR, Boulder, Colorado 80302

LANGUAGE FORTRAN

HISTORY This routine is from the IBM SSP Library.

ALGORITHM It uses Newton's iteration method. For references see page 4.

SPACE REQUIRED The routine RTNI requires 151₈.

ACCURACY Dependent on EPS.

TIMING Variable, depending on the complexity of the function F(X), and on the number of iterations. For 1000 iterations on a simple cubic function the timing of the iteration loop was approximately 98 milliseconds on the 6600.

PORTABILITY The routine RTNI is portable.

REQUIRED RESIDENT ROUTINES ULIBER

Method Subroutine RTNI is used to refine the initial guess X_0 of a root of the general nonlinear equation $f(X)=0$. Newton's iteration scheme is used in the following form:

$$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)} \quad (i = 0, 1, 2, \dots)$$

Method
(continued)

Each iteration step requires one evaluation of $f(X)$ and one evaluation of $f'(X)$. This iterative procedure is terminated if the following two conditions are satisfied:

$$\delta \leq \epsilon \text{ and } |f(X_{i+1})| \leq 100 \cdot \epsilon$$

with

$$\delta = \begin{cases} \left| \frac{X_{i+1} - X_i}{X_{i+1}} \right| & \text{in case of } |X_{i+1}| > 1 \\ |X_{i+1} - X_i| & \text{in case of } |X_{i+1}| \leq 1 \end{cases}$$

and tolerance ϵ given by input.

If there is no convergence the reasons may be the following:

1. Too few iteration steps are specified.
2. The initial guess X_0 is too far away from any root.
3. The tolerance ϵ is too small with respect to round off errors.
4. The root to be determined is of multiplicity greater than one.

The procedure also fails if at any iteration step the derivative $f'(X_i)$ becomes zero.

References

- [1] F.B. Hildebrand, 1956: "Introduction to Numerical Analysis," McGraw-Hill, New York/Toronto/London, pp. 447-450.
- [2] R. Zurmühl, 1963: "Praktische Mathematik für Ingenieure und Physiker," Springer, Berlin/Göttingen/Heidelberg, pp. 12-17.

INTERPOLATION/APPROXIMATION/SMOOTHING

BSL1NT

BSL2NT

CUBSPL

CURV

HRM1NT

HRM2NT

KURV

KURVP

QURV

SPLPAK

SURF

TRIANGLE

SUBROUTINE BSLINT (KFCN,KORD,X,F,MX,U,G,IU)

DIMENSION OF ARGUMENTS X(MX),F(MX),U(IU),G(IU)

LATEST REVISION November 1, 1973

PURPOSE To provide a general purpose routine for interpolating values and/or derivatives of a real function of one variable, given only its values at a sequence of tabular ordinates.

NB

If both function values and derivatives are known, the companion library routine 'HRLINT' should be used.

ACCESS CARDS *FORTRAN,S=ULIB,N=BSLINT
*COSY

USAGE

1. General call
 CALL BSLINT (KFCN,KORD,X,F,MX,U,G,IU)
2. Single function value
 FCN = BSLIF (U,X,F,MX)
3. Single derivative value
 DRV = BSLLD (U,X,F,MX)

ARGUMENTS

On Input

KFCN

Interpolation function/derivative flag.

- = 0 Function value.
- = 1 First derivative.
- = 2 Second derivative.

KORD

Interpolation order flag.

- = 1 Linear (2 points).
- = 2 Quadratic (3 points).
- = 3 Cubic (4 points).

X

Vector of independent abscissas. These must be monotonic (either ascending or descending), but need not be regularly spaced.

F

Vector of known dependent function values corresponding to X.

MX

Dimension of X- and F-vectors. If signed negative, an equally spaced X-grid is assumed.

U

Vector of new ordinate arguments. No order is imposed but each U(I) must be contained in (X(1),X(MX)).

IU

Dimension of U- and G-vectors. If signed negative, an equally spaced U-grid is assumed. If MX is also negative, U and X are treated as simple indexical grids with equal extent, (i.e., U(1) = X(1) and U(IU) = X(MX) for convenience of interpolation of finite-difference meshes.

On Output G
Vector of desired function values or derivatives.

NOTES

- If only MX is negative, a regular X-grid with spacing $X(2) - X(1)$ is assumed, avoiding search overhead.
- If both MX and IU are negative, co-extensive indexical grids are assumed. In this case, both X- and U-vectors may be dummies.

ENTRY POINTS BSLINT, BSLIF, BSLID

COMMON BLOCKS COMMON/BSL1CM/LOG,KERR,L,M,N,P,MX,XP,D2FM,D2FP,EPS,ONE

(For definitions of these internal parameters, see BSLINT routine).

I/O None

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Jordan Towner Hastings, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY

Written and standardized October 1973.

ALGORITHM

Binary search (beginning at previously successful index) to locate new ordinate argument in tabular function grid, followed by Bessel interpolation of varying (first through third) order, involving at most four function values in the local neighborhood of the desired point. Center-corrected second-order accurate finite differences are used to obtain continuous second derivatives throughout the interior of the interpolated region; at the grid boundaries these derivatives are linearly extrapolated.

REFERENCES

Center-corrected Differences, Sundqvist & Veronis, A Finite-Difference Grid with Non-Constant Intervals, *Tellus Vol. 22, No.1* (Jan. 1970) pp 26-31.

Bessel Interpolation, Buckingham, R.A., Numerical Methods, Pitman & Sons, Bath, U.K. (1962) p.111 ff.

SPACE REQUIRED

1000₈

ACCURACY

Highly dependent on density and spacing of tabular values F(X). Generally at least 3 to 4 digits for cubic interpolation on any reasonable grid.

TIMING

Approximately 15 milliseconds per 100 interpolated values.

PORTABILITY

- Labelled COMMON preset by DATA statements.
- Machine dependent constants, EPS and ONE.
- Local variables presumed preserved between calls.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

SUBROUTINE BSL2NT (KFCN,KORD,X,Y,F,MX,NY,U,V,G,IU,JV)

**DIMENSION OF
ARGUMENTS**

X(MX),Y(NY),F(MX,NY),U(IU),V(JV),G(IU,JV)

LATEST REVISION

November 1, 1973

PURPOSE

To provide a general purpose routine for interpolating values and/or derivatives of a real function of two variables, given only its values at intersections of a pair of tabular co-ordinates.

NB

If both function values and derivatives are known, the companion library routine 'HRM2NT' should be used.

ACCESS CARDS

*FORTRAN,S=ULIB,N=BSL2NT
*COSY

USAGE

1. General call

CALL BSL2NT (KFCN,KORD,X,Y,F,MX,NY,U,V,G,IU,JV)

2. Single function value

FCN = BSL2F (U,V,X,Y,F,MX,NY)

3. Single directional derivative

DRV = BSL2DX (U,V,X,Y,F,MX,NY)
DRV = BSL2DY (U,V,X,Y,F,MX,NY)

4. Interpolation/densification of function values between two equally-spaced (finite-difference) meshes.

CALL BSL2RG (F,MX,NY,G,IU,JV)

ARGUMENTS

On Input

KFCN

Interpolation function/derivative flag.

= -2 Second Y-derivative.

= -1 First Y-derivative.

= 0 Function value.

= +1 First X-derivative.

= +2 Second X-derivative.

KORD

Interpolation order flag.

= 1 Linear (2 points).

= 2 Quadratic (3 points).

= 3 Cubic (4 points).

X,Y

Vectors of independent co-ordinates. These must be monotonic (either increasing or decreasing), but need not be regularly spaced.

F

Matrix of known dependent function values corresponding to X,Y.

MX,NY

Dimensions of X- and Y-vectors respectively, and F-matrix. If signed negative, equally spaced X- and/or Y-grids are assumed.

U,V

Vectors of new co-ordinate arguments. No order is imposed, but each pair U(I),V(J) must be contained in (X(1),X(MX)), (Y(1),Y(NY)) respectively.

IU,JV

Dimensions of U- and V-vectors respectively, and G-matrix. If signed negative, equally spaced U- and/or V-grids are assumed. If MX and/or NY are also negative, U and X and/or V and Y are treated as simple indexical grids with equal extent, (i.e., $U(1) = X(1)$, $U(IU) = X(MX)$ and/or $V(1) = Y(1)$, $V(JV) = Y(NY)$) for convenience of interpolation between finite-difference meshes.

On Output**G**

Matrix of desired function values or derivatives.

NOTES

- If only MX and NY are negative, regular X- and Y-grids with spacing $X(2)-X(1)$ and $Y(2)-Y(1)$ are assumed, avoiding search overhead.
- If both MX and NY, and IU and JV are negative, co-extensive indexical grids are assumed. In this case, all X-, Y-, U- and V-vectors may be dummies.

ENTRY POINTS

BSL2NT, BSL2F, BSL2DX, BSL2DY, BSL2RG

COMMON BLOCKS

COMMON/BSL1CM/LOG,KERR,L,M,N,P,XM,XP,D2FM,D2FP,EPS,ONE

(For definitions of these internal parameters, see BSL1NT routine).

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

BSL1NT

SPECIALIST

Jordan Towner Hastings, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Written and standardized October 1973.

ALGORITHM

Repeated application of 1-dimensional Bessel Interpolation to construct local tensor products of approximating polynomials, involving at most 16 function values in the 4 grid squares most closely surrounding the desired point. (For further discussion of this method, see 'BSL1NT').

SPACE REQUIRED	1200 ₈ + BSL1NT
ACCURACY	Highly dependent on density and spacing of tabular values F(X,Y). Generally at least 3 to 4 digits for cubic interpolation on any reasonable grid.
TIMING	Approximately 50 milliseconds per 100 interpolated values on CDC 6600.
PORTABILITY	See BSL1NT
REQUIRED RESIDENT ROUTINES	ULIBER

PACKAGE CUBSPL**LATEST REVISION**

February 1974

PURPOSE

To perform one and two-dimensional cubic spline interpolation with choice of boundary conditions. The function and selected derivatives may be evaluated at any point where interpolation is required. In the two-dimensional case, the given data points must be on a rectangular grid, which need not be equally spaced. The package CUBSPL contains seven routines.

Subroutine COEFF1

Computes the coefficients for one-dimensional cubic spline interpolation using one of two boundary conditions at each end of the range.

- Second derivatives given at boundary.
- First derivative given at boundary.
- Periodic boundary condition.
- First derivative determined by fitting a cubic to the four points nearest to the boundary.

PURPOSE
(continued)

Subroutine TERP1

Using the coefficients computed by COEFF1, this routine evaluates the function and/or first and second derivatives at any point where interpolation is required.

Subroutine COEFF2

Computes the coefficients for two-dimensional bicubic spline interpolation with the same choice of boundary conditions as for COEFF1.

Function TERP2

Using the coefficients produced by COEFF2, this subroutine evaluates the function or a selected derivative at any point where two-dimensional interpolation is required.

Subroutine TRIP

A simple periodic, tridiagonal linear equation solver used by COEFF1.

Subroutine SEARCH

Performs a binary search in a one dimensional floating point table arranged in ascending order. This routine is called by TERP1 and TERP2.

Subroutine INTERP

Given coefficients provided by COEFF1 and the position of the interpolation point in the independent variable

table, this routine performs one-dimensional interpolation for the function value, first and second derivative, as desired. This routine is called by TERP1 and TERP2.

ACCESS CARDS	*FORTRAN,S=ULIB,N=CUBSPL *COSY
	These cards access all seven routines of the package, CUBSPL.
SPACE REQUIRED	Total space for all seven routines is $2243_8 = 1187_{10}$
ENTRY POINTS	COEFF1, TERP1 , COEFF2, TERP2 , TRIP, SEARCH, INTERP
SPECIAL CONDITIONS	Tables of independent variables must be arranged in ascending order. For two-dimensional interpolation, the data points must lie on a rectangular mesh.
COMMON BLOCKS	None
I/O	None
PRECISION	Single
REQUIRED ULIB ROUTINES	None

SPECIALIST

Cicely Ridley, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

This package is based on the routines

LA E 102A, SPL1D1

LA E 103A, SPL1D2

LA E 104A, SPL2D1

LA E 105A, SPL2D2

of the Los Alamos cubic spline package written by Thomas J. Jordan and Bertha Fagan, 1968. The routines have been streamlined and standardized. The algorithm for two-dimensional interpolation is considerably modified.

ALGORITHM

For one-dimensional interpolation, the cubic spline is expressed in terms of the function values and second derivatives at the data points. The second derivatives are determined from a tridiagonal linear system which describes the continuity of the first derivative and incorporates the given boundary conditions. COEFF1 sets up this system and calls subroutine TRIP to solve it.

The cubic segment between two adjacent tabular points is constructed from the function values and second derivatives at these points. These provide the four constants needed to define the cubic uniquely. From this cubic, values of the function and its first and second derivatives are readily determined at any intermediate point. One-dimensional interpolation is performed by the routine TERP1.

For two-dimensional interpolation, the bicubic spline is described in terms of values of F , F_{XX} , F_{YY} , and F_{XXYY} at each point on the given two-dimensional rectangular grid of data points. Here F is the function value,

$$F_{XX} = (D/DX)^{**2}*F$$

and so on. The coefficients are determined by $COEFF2$, which uses successive applications of $COEFF1$.

1. The array F_{XX} is determined by applying $COEFF1$ to F along each line in the X-direction.
2. The array F_{YY} is determined by applying $COEFF1$ to F along each line in the Y-direction.
3. F_{XXYY} is determined on the constant Y boundaries by applying $COEFF1$ to F_{YY} along these boundaries.
4. The remainder of the array F_{XXYY} is determined by applying $COEFF1$ to F_{XX} along each line in the Y-direction.

The bicubic within any rectangular element of the grid is constructed from the values of F , F_{XX} , F_{YY} , F_{XXYY} at the four corners. These provide the 16 constants necessary to define the bicubic uniquely. To find the value of F corresponding to a point (X_B, Y_B) within the elementary rectangle, $(X(I), Y(J))$, $(X(I+1), Y(J))$, $(X(I), Y(J+1))$, $(X(I+1), Y(J+1))$, five one dimensional interpolations are performed.

1. F at $(X_B, Y(J))$ and $(X_B, Y(J+1))$ are found by interpolating F in the X-direction using F_{XX} . (Two interpolations).

ALGORITHM
(continued)

2. FYY at (XB, Y(J)) and (XB, Y(J+1)) are found by interpolating FYY in the X-direction using FXXYY. (Two interpolations.)
3. Finally F at (XB, YB) is found by interpolating between F(XB, Y(J)) and F(XB, Y(J+1)) in the Y-direction using values of FYY(XB, Y(J)) and FYY(XB, Y(J+1)) obtained above. (One interpolation.)

Two-dimensional interpolation is performed in TERP2.

For greater detail, see J. L. Walsh, J. H. Ahlberg, E. N. Nilsen, "Best Approximation Properties of the Spline Fit," Journal of Mathematics and Mechanics, Vol. II (1962), 225-234.

T. L. Jordan, "Smoothing and Multivariable Interpolation with Splines," Los Alamos Report, LA-3137, 1965.

ACCURACY

Near machine accuracy was obtained when interpolating a cubic in one dimension or a bicubic in two dimensions.

PORTABILITY

Standards are satisfied.

**REQUIRED RESIDENT
ROUTINES**

None

SUBROUTINE COEFF1 (N,X,F,W,IOP,INT,WK)

DIMENSION OF ARGUMENTS X(N),F(INT*(N-1)+1),W(INT*(N-1)+1),IOP(2),WK(3*N+1)

LATEST REVISION February 1974

USAGE CALL COEFF1 (N,X,F,W,IOP,INT,WK)

ARGUMENTS

On Input

N

The number of data points. N must be at least 4.

X

Table of N independent variable values in ascending order. Dimension of X in calling program must be at least N.

F

Table of N corresponding dependent variable values. The values are separated by interval INT. This is usually unity for one-dimensional interpolation. Other values are useful when COEFF1 is incorporated in a two-dimensional interpolation scheme (see COEFF2). Dimension of F in the calling program must be at least (INT*(N-1)+1).

IOP

Two element integer array defining boundary conditions at X(1) and X(N) according to the following code. For IOP(1)

= 1 Second derivative given at X(1). Place value of second derivative in W(1) before call to COEFF1.

On Input
(continued)

- = 2 First derivative given at X(1). Place value of first derivative in W(1) before call.
- = 3 Periodic boundary condition. X(1) and X(N) are equivalent points. F(1) and F(INT*(N-1)+1) are equal.
- = 4 The first derivative at X(1) is calculated by fitting a cubic to points X(1) through X(4).

Similarly, IOP(2) defines the boundary condition at X(N). When IOP(2)=1 (or 2), the value of the second (or first) derivative must be placed in W(INT*(N-1)+1). Note that if IOP(1)=3, consistency demands that IOP(2)=3 also.

INT

Spacing in F and W tables. For one-dimensional interpolation this will usually be unity.

WK

Work area of dimension at least (3*N+1).

On Output

W

Table of second derivatives corresponding to given X and Y values. The separation of tabular entries is INT (see above). Dimension of W in the calling program must be at least (INT*(N-1)+1).

The arrays X, F, W are used as input for the routine TERPL, which performs interpolation at a given value of the independent variable.

SPACE REQUIRED

$$654_8 = 428_{10}$$

TIMING

The timing is linearly proportional to N, the number of data points. For 21 data points, the time on the NCAR CDC 7600 was approximately .24 milliseconds.

SUBROUTINE TERP1 (N,X,F,W,Y,INT,TAB,ITAB)**DIMENSION OF ARGUMENTS**

X(N),F(INT*(N-1)+1),W(INT*(N-1)+1),TAB(3),ITAB(3)

LATEST REVISION

February 1974

USAGE

CALL TERP1 (N,X,F,W,Y,INT,TAB,ITAB)

ARGUMENTS**On Input**

N

The number of data points. N must be at least 4.

X

Table of N independent variable values in ascending order. Dimension of X in the calling program must be at least N.

F

Table of N corresponding dependent variable values separated by interval INT, usually unity for one-dimensional interpolation. Dimension of F in the calling program must be at least (INT*(N-1)+1).

W

Table of second derivatives computed by COEFF1. The separation of tabular entries is INT. Dimension of W in the calling program must be at least (INT*(N-1)+1).

Y

Value of the independent variable at which interpolation is required. If Y lies outside the range of the table, extrapolation takes place.

On Input*(continued)***INT**

Spacing of tabular entries in F and W arrays. This is usually unity for one-dimensional interpolation.

ITAB

Three element integer array defining interpolation to be performed at Y.

If ITAB(1)=1, the function value is returned in TAB(1).

If ITAB(2)=1, the first derivative is returned in TAB(2).

If ITAB(3)=1, the second derivative is returned in TAB(3).

If ITAB(I)=0 for I=1, 2 or 3, the corresponding function value or derivative is not computed and TAB(I) is not referenced.

On Output**TAB**

Three element array in which interpolated function value, first and second derivatives are returned as dictated by ITAB (see above).

SPACE REQUIRED $47_8 = 39_{10}$ **TIMING**

This procedure is fast. The maximum time for the binary search is proportional to $A \log(N)$. The time for function and derivative evaluation is independent of N. For 21 data points the time taken on the NCAR 7600 is about 80 microseconds.

SUBROUTINE COEFF2 (NX,X,NY,Y,F,FXX,FYY,FXXYY,IDM,IBD,WK)

**DIMENSION OF
ARGUMENTS**

X(NX),Y(NY),F(IDM,NY),FXX(IDM,NY),FYY(IDM,NY),FXXYY(IDM,NY),
IBD(4),WK(3*MAXO(NX,NY)+1)

(IDM must be .GE. NX)

LATEST REVISION

February 1974

USAGE

CALL COEFF2 (NX,X,NY,Y,F,FXX,FYY,FXXYY,IDM,IBD,WK)

ARGUMENTS

On Input

NX

Number of grid points in the X-direction. NX must be at least 4.

X

Table of NX values of the first independent variable arranged in ascending order. Dimension of X in the calling program must be at least NX.

NY

Number of grid points in the Y-direction. NY must be at least 4.

Y

Table of NY values of the second independent variable arranged in ascending order. Dimension of Y in the calling program must be at least NY.

On Input
(continued)

F

Two dimensional array of function values at the grid points defined by the arrays X and Y. Dimension of F in the calling program is (IDM, NYY) where

IDM.GE.NX
NYY.GE.NY

IDM

First dimension in the calling program of arrays F, FXX, FYY, FXXYY. IDM must be at least NX.

IBD

Four element integer array defining boundary conditions according to the following code. For IBD(1)

- = 1 The second derivative of F with respect to X is given at (X(1),Y(J)) for J=1, NY, 1. Values of this second derivative must be placed in FXX(1,J) for J=1, NY, 1, before calling COEFF2.
- = 2 The first derivative of F with respect to X is given at (X(1), Y(J)) for J=1, NY, 1. Values of the derivative must be placed in FXX(1,J) for J=1, NY, 1 before calling COEFF2.
- = 3 Periodic boundary condition in the X-direction. (X(1), Y(J)) and (X(NX), Y(J)) are equivalent points for J=1, NY, 1. F(1,J) and F(NX,J) are equal.
- = 4 The first derivative of F with respect to X at (X(1), Y(J)) is computed by fitting a cubic to F(1,J) through F(4,j) for J=1, NY, 1.

Similarly, IBD(2) defines the boundary condition at (X(NX),Y(J)) for J=1, NY, 1. When IBD(2)=1 (or 2) the values of the second (or first) derivative of F with respect to X are placed in FXX(NX,J) for J=1, NY, 1. Note that if IBD(1)=3, consistency requires that IBD(2)=3 also. For IBD(3)

- = 1 The second derivative of F with respect to Y is given at (X(I), Y(1)). Place values of the derivative in FYY(I,1) for I=1, NX, 1 before calling COEFF2.
- = 2 The first derivative of F with respect to Y is given at (X(I), Y(1)). Values of this derivative must be placed in FYY(I,L) for I=1, NX, 1 before calling COEFF2.

- = 3 Periodic boundary condition in the Y-direction. (X(I),Y(1)) and (X(I),Y(NY)) are equivalent points. F(I,1) and F(I,NY) are equal.
- = 4 The first derivative of F with respect to Y at (X(I),Y(1)) is computed by fitting a cubic to F(I,1) through F(I,4) for I=1,NX,1.

Similarly, IBD(4) defines the boundary condition at (X(I),Y(NY)) for I=1,NX,1 and given derivative values are placed in FYY(I,NY). Note that consistency demands that if IBD(3)=3, then IBD(4)=3 also.

WK

Work area of dimension at least (3*MAXO(NX,NY)+1)

On Output

FXX

Array of second derivatives of F with respect to X computed by COEFF2. FXX(I,J) is derivative at (X(I),Y(J)). As for F, dimension of FXX in the calling program is (IDM,NYY).

FYY

Array of second derivatives of F with respect to Y computed by COEFF2. Dimension of FYY in the calling program is (IDM,NYY).

FXXYY

Array of fourth derivatives $(D/DX)^{**2}*(D/DY)^{**2}*F$, computed by COEFF2. Dimension of FXXYY in the calling program is (IDM,NYY).

The arrays X, Y, F, FXX, FYY, FXXYY are used as input for the routine TERP2 which performs interpolation at required values of the two independent variables.

SPACE REQUIRED

370₈ = 248₁₀

TIMING

The timing is proportional to $NX*NY$. For a 21×21 grid of data points, the time taken on the NCAR 7600 was 19.5 milliseconds.

FUNCTION TERP2 (XB,YB,NX,X,NY,Y,F,FXX,FYY,FXXYY,IDM,IXD,IYD)**DIMENSION OF ARGUMENTS**

X(NX),Y(NY),F(IDM, NY),FXX(IDM,NY),FYY(IDM,NY),FXXYY(IDM,NY))
 (IDM must be .GE. NX)

LATEST REVISION

February 1974

USAGE

R= TERP2 (XB,YB,NX,X,NY,Y,F,FXX,FYY,FXXYY,IDM,IXD,IYD)

ARGUMENTS**On Input**

XB, YB

Values of the independent variables, X and Y, at which interpolation is required.

NX

Number of grid points in the X-direction. NX must be at least 4.

X

Table of NX values of the independent variable, X, arranged in ascending order. Dimension of X in the calling program must be at least NX.

NY

Number of grid points in the Y-direction. NY must be at least 4.

Y

Table of NY values of the independent variable, Y, arranged in ascending order. Dimension of Y in the calling program must be at least

On Input
(continued)

F

Two-dimensional array of function values at grid points defined by the arrays X and Y.

Dimension of F in the calling program is (IDM,NYY), where

IDM.GE.NX
NYY.GE.NY

FXX

Array of second derivatives of F with respect to X computed by COEFF2. Dimension of FXX in the calling program is (IDM,NYY). See under F above.

FYY

Array of second derivatives of F with respect to Y computed by COEFF2. Dimension of FYY in the calling program is (IDM,NYY).

FXXYY

Array of fourth derivatives, $(D/DX)^{**2}*(D/DY)^{**2}*F$, computed by COEFF2. Dimension of FXXYY in the calling program is (IDM,NYY).

IDM

First dimension in the calling program of arrays F, FXX, FYY and FXXYY,

IDM.GE.NX

IXD, IYD

Define derivative to be returned by the function INTRP2. IXD, IYD may each take the values 0, 1, 2. The derivative returned is $(D/DX)^{**IXD}*(D/DY)^{**IYD}*F$. Note that if IXD=IYD=0, the function value itself is returned.

SPACE REQUIRED

$220_8 = 144_{10}$

TIMING

This procedure is fast. The maximum time for the binary search is proportional to $\text{ALOG}(\text{NX}*\text{NY})$. The time for function evaluation is independent of N. For a 21×21 grid of data points, an average time for an interpolation on the NCAR CDC 7600 is about .29 milliseconds.

PACKAGE CURV**LATEST REVISION** October 1973**PURPOSE**

To perform interpolation, differentiation, and integration using splines under tension through a sequence of functional values. The functional values may be given at arbitrary abscissas and the slopes at the two ends of the curve may be specified or omitted. The package CURV contains four routines.

Subroutine CURV1

Determines the parameters necessary to compute an interpolatory spline under tension through a sequence of functional values. The slopes at the two ends of the curve may be specified or omitted.

Function CURV2

Interpolates a curve at a given point using a spline under tension determined by the parameters calculated in CURV1.

PURPOSE

(continued)

Function CURVD

Differentiates a curve at a given point using an interpolatory spline under tension determined by the parameters calculated in CURV1.

Function CURVI

Integrates a curve between two given limits using an interpolatory spline under tension determined by the parameters calculated in CURV1.

ACCESS CARDS

*FORTRAN,S=ULLIB,N=CURV

*COSY

These cards access all four routines of the package CURV.

SPACE REQUIRED

Total space for all four routines is $1307_8 = 711_{10}$

ENTRY POINTS

CURV1, CURV2, CURVD, CURVI

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Russell K. Rew, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

This package was originally written by Alan K. Cline in March 1972 (CURV1 and CURV2) and August 1973 (CURVD and CURVI). The routines have been standardized without modifying the algorithms.

ALGORITHM

In CURV1, a tridiagonal linear system is formed and solved for a vector YP of values proportional to second derivatives of the curve at the abscissas X_i , $i = 1, 2, \dots, N$.

In CURV2, if T is the value to be mapped onto the interpolating curve, an interval (X_{i-1}, X_i) is determined containing T . For $T < X_1$, $i = 2$ is used, and for $T > X_N$, $i = N$ is used. The interpolation is performed using the formula

$$\begin{aligned} \text{CURV2}(T, \dots) = & (Y_P_i \sinh(\sigma(T - X_{i-1})) + Y_{P_{i-1}} \sinh(\sigma(X_i - T))) \\ & / \sinh(\sigma(X_i - X_{i-1})) + ((Y_i - Y_{P_i})(T - X_{i-1}) \\ & + (Y_{i-1} - Y_{P_{i-1}})(X_i - T)) / (X_i - X_{i-1}) \end{aligned}$$

In CURVD, the differentiation of the spline is performed using the formula

$$\begin{aligned} \text{CURVD}(T, \dots) = & \sigma(Y_{P_i} \cosh(\sigma(T - X_{i-1})) + Y_{P_{i-1}} \cosh(\sigma(X_i - T))) / \\ & \sinh(\sigma(X_i - X_{i-1})) + ((Y_i - Y_{P_i}) - (Y_{i-1} - Y_{P_{i-1}})) / (X_i - X_{i-1}) \end{aligned}$$

In CURVI, the spline under tension passing through the given data is integrated analytically using the formula

$$\begin{aligned} \text{CURVI}(a,b,\dots) = & (Y_{P_i}(\cosh(\sigma(b-X_{i-1})) - \cosh(\sigma(a-X_{i-1}))) \\ & - Y_{P_{i-1}}(\cosh(\sigma(X_i-b)) - \cosh(\sigma(X_i-a))))/\sinh(\sigma(X_i-X_{i-1})) \\ & + ((Y_i - Y_{P_i}) \cdot ((b-X_{i-1})^2 - (a-X_{i-1})^2) - (Y_{i-1} - Y_{P_{i-1}}) \\ & \cdot ((X_i-b)^2 - (X_i-a)^2))/(2 \cdot (X_i - X_{i-1})) \end{aligned}$$

in each of the subintervals of the form (X_{i-1}, X_i) from XL to XU.

For greater detail, see

Cline, A.K. "Scalar- and Planar-Valued Curve Fitting Using Splines Under Tension," CACM 17, 4 pp. 218-223.

ACCURACY

Nearly full precision for σ (tension factor) of about one in magnitude. For $|\sigma| < .001$ accuracy may seriously degenerate due to SINH approximation.

PORTABILITY

Fully portable.

REQUIRED RESIDENT ROUTINES

EXP

SUBROUTINE CURV1 (N,X,Y,SLP1,SLPH,YP,TEMP,SIGMA)

DIMENSION OF ARGUMENTS X(N),Y(N),YP(N),TEMP(N)

LATEST REVISION October 22, 1973

USAGE CALL CURV1 (N,X,Y,SLP1,SLPN,YP,TEMP,SIGMA)

ARGUMENTS**On Input**

N

Is the number of values to be interpolated (N.GE.2).

X

Is an array of the N increasing abscissas of the functional values.

Y

Is an array of the N ordinates of the values, (i.e., Y(K) is the functional value corresponding to X(K)).

SLP1 and SLPN

Contain the desired values for the first derivative of the curve at X(1) and X(N), respectively. If the quantity SIGMA is negative these values will be determined internally and the user need only furnish place-holding (dummy) parameters for SLP1 and SLPN. Such place-holding parameters will be ignored but not destroyed.

YP

Is an array of length at least N.

On Input
(continued)

TEMP

Is an array of length at least N which is used for scratch storage.

SIGMA

Contains the tension factor. This is non-zero and indicates the curviness desired. If ABS(SIGMA) is nearly zero (e.g., .001) the resulting curve is approximately a cubic spline. If ABS(SIGMA) is large (e.g., 50.) the resulting curve is nearly a polygonal line. The sign of SIGMA indicates whether the derivative information has been input or not. If SIGMA is negative the endpoint derivatives will be determined internally. A standard value for SIGMA is approximately 1. in absolute value.

On Output

YP

Contains values proportional to the second derivative of the curve at the given nodes.

N, X, Y, SLP1, SLPN and SIGMA are unaltered.

SPACE REQUIRED

$326_8 = 214_{10}$ locations

TIMING

Proportional to N. The approximate running time for N=50 on the NCAR CDC 7600 is .6 millisecond.

FUNCTION CURV2 (T,N,X,Y,YP,SIGMA,IT)**DIMENSION OF ARGUMENTS**

X(N),Y(N),YP(N)

LATEST REVISION

October 22, 1973

USAGE

YT = CURV2 (T,N,X,Y,YP,SIGMA,IT)

ARGUMENTS**On Input**

T

Contains a real value to be mapped onto the interpolating curve.

N

Contains the number of points which were interpolated to determine the curve.

X and Y

Are arrays containing the abscissas and ordinates of the interpolated points.

YP

Is an array with values proportional to the second derivative of the curve at the nodes.

SIGMA

Contains the tension factor (its sign is ignored).

On Input
(continued)

IT

Is an integer switch. If IT is not 1, this indicates that the function has been called previously (with N, X, Y, YP and SIGMA unaltered) and that this value of T exceeds the previous value. With such information the function is able to perform the interpolation much more rapidly. If a user seeks to interpolate at a sequence of points, efficiency is gained by ordering the values increasing and setting IT to the index of the call. If IT is 1 the search for the interval (X(K),X(K+1)) containing T starts with K=1.

The parameters N, X, Y, YP and SIGMA should be input unaltered from the output of CURV1.

On Output

CURV2

Contains the interpolated value. For T less than X(1), CURV2=Y(1). For T greater than X(N), CURV2=Y(N).

None of the input parameters are altered.

SPACE REQUIRED

150₈ = 104₁₀ locations

TIMING

In efficient mode (IT≠1), the running time per call on the NCAR CDC 7600 is approximately 42 microseconds.

FUNCTION CURVD (T,N,X,Y,YP,SIGMA,IT)

DIMENSION OF ARGUMENTS X(N),Y(N),YP(N)

LATEST REVISION October 22, 1973

USAGE YTD = CURVD (T,N,X,Y,YP,SIGMA,IT)

ARGUMENTS**On Input**

T

Contains a real value at which the derivative is to be determined.

N

Contains the number of points which were interpolated to determine the curve.

X and Y

Arrays containing the abscissas and ordinates of the interpolated points.

YP

An array with values proportional to the second derivative of the curve at the nodes.

SIGMA

Contains the tension factor (its sign is ignored).

On Input
(continued)

IT

An integer switch. If IT is not 1 this indicates that the function has been called previously (with N, X, Y, YP and SIGMA unaltered) and that this value of T exceeds the previous value. With such information the function is able to perform the differentiation much more rapidly. If a user seeks to differentiate at a sequence of points, efficiency is gained by ordering the values increasing and setting IT to the index of the call. If IT is 1 the search for the interval (X(K),X(K+1)) containing T starts with K=1.

The parameters N, X, Y, UP and SIGMA should be input unaltered from the output of CURV1.

On Output

CURVD

Contains the derivative value.

None of the input parameters are altered.

SPACE REQUIRED

131₈ = 89₁₀ locations

TIMING

In efficient mode (IT≠1) running time per call on the NCAR CDC 7600 is 38 microseconds.

FUNCTION CURVI (XL,XU,N,X,Y,YP,SIGMA)

DIMENSION OF ARGUMENTS X(N),Y(N),YP(N)

LATEST REVISION October 22, 1973

USAGE YTI = CURVI (XL,XU,N,X,Y,YP,SIGMA)

ARGUMENTS**On Input**

XL and XU

Contain the lower and upper limits of integration, respectively. (XL need not be less than or equal to XU, $CURVI(XL,XU,\dots) = -CURVI(XU,XL,\dots)$).

N

Contains the number of points which were interpolated to determine the curve.

X and Y

Arrays containing the abscissas and ordinates of the ordinates of the interpolated points.

YP

An array with values proportional to the second derivative of the curve at the nodes.

SIGMA

Contains the tension factor (its sign is ignored).

The parameters N, X, Y, YP and SIGMA should be input unaltered from the output of CURV1.

On Output

CURVI

Contains the integral value.

None of the input parameters are altered.

SPACE REQUIRED

$460_8 = 304_{10}$ locations

TIMING

Proportional to N in most cases. For N=50 the running time on the NCAR CDC 7600 is 5.1 milliseconds.

SUBROUTINE HRMINT (KFCN,KORD,X,F,DF,MX,U,G,IU)**DIMENSION OF
ARGUMENTS**

X(MX),F(MX),D(MX,2),U(IU),G(IU)

LATEST REVISION

November 1973

PURPOSE

To provide a general purpose routine for interpolating values and/or derivatives of a real function of one variable, given both its values and first (and possibly second) derivative(s) at a sequence of tabular ordinates.

****NB****

If only function values, and not derivatives, are known, the companion library routine 'BSLLNT' should be used.

METHOD

Piecewise cubic, Hermite (or quintic, Hyperosculatory) polynomials, with continuous first (and second) derivatives.

ACCESS CARDS

*FORTRAN,S=ULIB,N=HRMINT
*COSY

USAGE

1. General Call

CALL HRMLNT (KFCN,KORD,X,F,DF,MX,U,G,IU)

2. Single function value

FCN = HRMIF (U,X,F,DF,MX)

3. Single derivative value

DRV = HRMID (U,X,F,DF,MX)

ARGUMENTS

On Input

KFCN

Interpolation function/derivative flag.

- = 0 Function value.
- = 1 First derivative.
- = 2 Second derivative.

KORD

Interpolation order flag.

- = 1 Linear (Function values only)
- = 3 Cubic (Function values and first derivative)
- = 5 Quintic (Function values and first and second derivatives)

X

Vector of independent abscissas. These must be monotonic (either ascending or descending), but need not be regularly spaced.

F

Vector of known dependent function values corresponding to X.

DF

Vector(s) of function derivative(s). If KORD = 1, this argument may be a dummy. Otherwise, DF(M,1) should contain first derivative of F(X(M,1)), and DF(M,2) should contain corresponding second derivatives, if KORD = 5.

MX

Dimension of X- and F-vectors. If signed negative, an equally spaced X-grid is assumed.

U

Vector of new ordinate arguments. No order is imposed but each U(I) must be contained in (X(1),X(MX)).

IU

Dimension of U- and G-vectors. If signed negative, an equally spaced U-grid is assumed. If MX is also negative, U and X are treated as simple indexical grids with equal extent (i.e., U(1) = X(1) and U(IU) = X(MX)) for convenience of interpolation of finite difference meshes.

On Output

G

Vector of desired function values or derivatives.

NOTES

- If only MX is negative, a regular X-grid with spacing X(2) - X(1) is assumed, avoiding search overhead.
- If both MX and IU are negative, co-extensive indexical grids are assumed. In this case, both X- and U-vectors may be dummies.

ENTRY POINTS

HRMLNT, HRMIF, HRMID

COMMON BLOCKS

COMMON/HRMLCM/LOG,KERR,L,M,N,P,XM,XP,D1XM,D1XP,D2XM,D2XP,
EPS,ONE

(For definitions of these internal parameters, see HRMLNT
routine.)

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Jordan Towner Hastings, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Written and standardized October 1973.

ALGORITHM

Binary search (beginning at previously successful index) to
locate new ordinate argument in tabular function grid,
followed by Hermite Interpolation of varying (first, third
or fifth) order, involving two function values and
corresponding first (and second) derivatives adjacent to
the desired point.

REFERENCES

Center-corrected Differences, Sunqvist & Veronis, A
Finite-Difference Grid with Non-Constant Intervals,
Tellus Vol. 22, No. 1 (Jan. 1970) pp 26-31.

Hermite Interpolation, Isaacson & Keller '*Analysis of
Numerical Methods*' Wiley, New York (1966) p 192-ff.

Hyperosculatory Interpolation, Salzer, Herbert E. '*Tables for
Hyperosculatory Interpolation*' General Dynamics, San Diego
(1962)

SPACE REQUIRED

1000₈

ACCURACY

Highly dependent on density and spacing of tabular values,
F(X). Generally, at least 3 to 4 digits for Hermite (cubic)
Interpolation, and 5 to 6 digits for Hyperosculatory (Quintic)
Interpolation, on any reasonable grid.

TIMING

Approximately 15 milliseconds per 100 Hermite Interpolated
values, 25 milliseconds per 100 Hyperosculatory Interpolated
values on the NCAR CDC 6600.

PORTABILITY

- Labelled COMMON preset by DATA statements.
- Machine dependent constants, EPS and ONE.
- Local variables presumed preserved between calls.

REQUIRED RESIDENT
ROUTINES

ULIBER

SUBROUTINE HRM2NT (KFCN,KORD,X,Y,F,DFX,DFY,U,V,G,IU,JV)**DIMENSION OF
ARGUMENTS**

X(MX),Y(NY),F(MX,NY),DX(MX,NY,2),DY(MX,NY,2),U(IU),V(JV),
G(IU,JV)

LATEST REVISION

November 1973

PURPOSE

To provide a general purpose routine for interpolating values and/or derivatives of a real function of two variables, given both its values and first (and possibly second) directional derivative(s) at intersections of a pair of tabular co-ordinates

****NB****

If only function values, and not derivatives, are known, the companion Library routine 'BSL2NT' should be used.

METHOD

Piecewise bi-cubic Hermite, or bi-quintic Hyperosculatory, polynomials, with continuous first and second derivatives.

ACCESS CARDS

*FORTRAN,S=ULIB,N=HRM2NT
*COSY

USAGE

1. General call

CALL HRM2NT (KFCN,KORD,X,Y,F,DFX,DFY,MX,NY,U,V,G,IU,JV)

2. Single function value

FCN = HRM2F (U,V,X,Y,F,DFX,DFY,MX,NY)

3. Single directional derivative

DRV = HRM2DX (U,V,X,Y,F,DFX,DFY,MX,NY)

DRV = HRM2DY (U,V,X,Y,F,DFX,DFY,MX,NY)

ARGUMENTS

On Input

KFCN

Interpolation function/derivative flag.

= -2 Second Y-derivative.

= -1 First Y-derivative.

= 0 Function value.

= +1 First X-derivative.

= +2 Second X-derivative.

KORD

Interpolation order flag.

= 1 Linear (Function values only)

= 3 Cubic (Function values and first derivative)

= 5 Quintic (Function values and first and second derivatives)

X,Y

Vectors of independent co-ordinates. These must be monotonic (either increasing or decreasing), but need not be regularly spaced.

DFX,DFY

Matrices of directional function derivatives. If KORD = 1, these arguments may be dummies. Otherwise, DFX(M,N,1) and DFY(M,N,1) should contain first directional derivatives of F(X(M),Y(N)) and DFX(M,N,2) and DFY(M,N,2) should contain corresponding second derivatives if KORD = 5.

F

Matrix of known dependent function values corresponding to X,Y.

MX,NY

Dimensions of X- and Y-vectors respectively, and F-matrix. If signed negative, equally spaced X- and/or Y-grids are assumed.

U,V

Vectors of new co-ordinate arguments. No order is imposed but each pair U(I), V(J) must be contained in (X(1),X(MX)), (Y(1),Y(NY)) respectively.

IU,JV

Dimensions of U- and V-vectors respectively, and G-matrix. If signed negative, equally spaced U- and/or V-grids are assumed. If MX and/or NY are also negative, U and X and/or V and Y are treated as simple indexical grids with equal extent (i.e., U(1) = X(1), U(IU) = X(MX) and/or V(1) = Y(1), V(JV) = Y(NY)) for convenience of interpolation between finite difference meshes.

On Output

G

Matrix of desired function values or derivatives.

NOTES

- If only MX and NY are negative, regular X- and Y- grids with spacing $X(2) - X(1)$ and $Y(2) - Y(1)$ are assumed, avoiding search overhead.

NOTES

(continued)

- If both MX and NY, and IU and JV are negative, co-extensive indexical grids are assumed. In this case, all X-, Y-, U- and V-vectors may be dummies.

ENTRY POINTS

HRM2NT, HRM2F, HRM2DX, HRM2DY

COMMON BLOCKS

COMMON/HRM1CM/LOG,KERR,L,M,N,P,XM,XP,D1XM,D1XP,D2XM,
D2XP,EPS,ONE

(For definitions of these internal parameters, see HRM1NT routine).

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINE**

HRM1NT

SPECIALIST

Jordan Towner Hastings, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Written and standardized October 1973.

ALGORITHM

Repeated application of 1-dimensional Hermite Interpolation to construct local tensor products of approximating polynomials involving the four function values and associated derivatives in the grid square immediately surrounding the desired point.

SPACE REQUIRED

1200₈ + HRMLNT

ACCURACY

Highly dependent on density and spacing of tabular values, F(X). Generally, at least 3 to 4 digits for Hermite (cubic) Interpolation, and 5 to 6 digits for Hyperoscillatory (Quintic) Interpolation, on any reasonable grid.

TIMING

Approximately 50 milliseconds per 100 Hermite Interpolated values, 100 milliseconds per 100 Hyperoscillatory Interpolated values on the NCAR CDC 6600.

PORTABILITY

See HRMLNT

**REQUIRED RESIDENT
ROUTINE**

ULIBER

PACKAGE KURV**LATEST REVISION**

October 1973

PURPOSE

To perform the mapping of points in the interval $(0.,1.)$ onto a curve in the plane, using splines under tension passing through a sequence of pairs $(X(1),Y(1)), \dots, (X(N),Y(N))$. The slopes at the two ends of the curve may be specified or omitted. The package KURV contains two routines.

Subroutine KURV1

KURV1 determines the parameters necessary to compute a spline under tension passing through a sequence of pairs $(X(1),Y(1)), \dots, (X(N),Y(N))$ in the plane. The slopes at the two ends of the curve may be specified or omitted.

Subroutine KURV2

KURV2 performs the mapping of points in the interval $(0.,1.)$ onto a curve in the plane. The subroutine KURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric

2.KURV.2

PURPOSE representation both of whose components are splines under
(continued) tension and functions of the polygonal arclength parameter.

ACCESS CARDS *FORTRAN,S=ULIB,N=KURV
*COSY

These cards access both routines of the package, KURV.

SPACE REQUIRED Total space for both routines is $735_8 = 477_{10}$.

ENTRY POINTS KURV1, KURV2

SPECIAL CONDITIONS None

COMMON BLOCKS None

I/O None

PRECISION Single

**REQUIRED ULIB
ROUTINES** None

SPECIALIST Russell K. Rew, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Originally written by A.K. Cline, March 1972.

ALGORITHM A tridiagonal system is formed and solved for the vectors XP and YP of values proportional to the curvature of the curve at the data points.

Curve values are mapped in a two stage process. First the value $ST = T * S$ of polygonal arclength is placed in an interval (S_{i-1}, S_i) where S_i is the polygonal arclength from the first data point to the i^{th} point. Then spline under tension interpolation is done in the two coordinate directions as functions of ST.

ACCURACY Nearly full precision for SIGMA of about one in magnitude. For $ABS(SIGMA)$ less than .001 accuracy may seriously degenerate due to SINH approximation.

PORTABILITY Fully portable (the 17 digit representation of PI in the first executable statement of routine KURV1 may be altered, if desired.)

REQUIRED RESIDENT ROUTINES EXP, ATAN2, COS, SIN, SQRT

SUBROUTINE KURV1 (N,X,Y,SLP1,SLPN,XP,YP,TEMP,S,SIGMA)

DIMENSION OF ARGUMENTS X(N),Y(N),XP(N),YP(N),TEMP(N)

LATEST REVISION October 22, 1973

USAGE CALL KURV1 (N,X,Y,SLP1,SLPN,XP,YP,TEMP,S,SIGMA)

ARGUMENTS

On Input

N

Is the number of points to be interpolated (N.GE.2).

X

Is an array containing the N X-coordinates of the points.

Y

Is an array containing the N Y-coordinates of the points.

SLP1 and SLPN

Contain the desired values for the slope of the curve at (X(1),Y(1)) and (X(N),Y(N)), respectively. These quantities are in degrees and measured counter-clockwise from the positive X-axis. The positive sense of the curve is assumed to be that moving from the point 1 to point N. If the quantity SIGMA is negative these slopes will be determined internally and the user need only furnish place-holding (dummy) parameters for SLP1 and SLPN. Such place-holding parameters will be ignored but not destroyed.

On Output

XP and YP

Are arrays of length at least N.

On Output
(continued)

S
Contains the polygonal arclength of the curve.

N, X, Y, SLP1, SLPN and SIGMA are unaltered.

SPACE REQUIRED

$530_8 = 344_{10}$ locations

TIMING

The timing is linearly proportional to N, the number of data points. For 21 data points, the time on the NCAR CDC 7600 is approximately .24 millisecond.

SUBROUTINE KURV2 (T,XS,YS,N,X,Y,XP,YP,S,SIGMA)

DIMENSION OF ARGUMENTS X(N),Y(N),XP(N),YP(N)

LATEST REVISION October 22, 1973

USAGE CALL KURV2 (T,XS,YS,N,X,Y,XP,YP,S,SIGMA)

ARGUMENTS

On Input

T

Contains a real value of absolute value less than or equal to 1. to be mapped to a point on the curve. The sign of T is ignored and the interval (0.,1.) is mapped onto the entire curve. If T is negative, this indicates that the subroutine has been called previously (with all other input variables unaltered) and that this value of T exceeds the previous value in absolute value. With such information the subroutine is able to map a sequence of points onto the same curve, efficiency is gained by ordering the values increasing in magnitude and setting the signs of all but the first negative.

N

Contains the number of points which were interpolated to determine the curve.

X and Y

Arrays containing the X- and Y-coordinates of the interpolated points.

XP and YP

Are the arrays output from KURV1 containing curvature information.

On Input
(continued)

S
Contains the polygonal arclength of the curve.

SIGMA
Contains the tension factor (its sign is ignored).

The parameters N, X, Y, XP, YP, S and SIGMA should be input unaltered from the output of KURV1.

On Output

XS and YS
Contain the X- and Y-coordinates of the image point on the curve.

T, N, X, Y, XP, YP, S and SIGMA are unaltered.

SPACE REQUIRED

$205_8 = 133_{10}$ locations

TIMING

In efficient mode ($T < 0.$) running time per call on the NCAR CDC 7600 is 55 microseconds.

PACKAGE KURVP**LATEST REVISION** October, 1973

PURPOSE To perform the mapping of points in the interval (0.,1.) onto a *closed* curve in the plane, using splines under tension passing through a sequence of pairs (X(1),Y(1)),..., (X(N),Y(N)).

Subroutine KURVP1

KURVP1 determines the parameters necessary to compute a spline under tension forming a closed curve in the plane and passing through a sequence of pairs (X(1),Y(1)),... (X(N),Y(N)). For actual computation of points on the curve it is necessary to call the subroutine KURVP2.

Subroutine KURVP2

KURVP2 performs the mapping of points in the interval (0.,1.) onto a closed curve in the plane. The subroutine KURVP1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation both of whose components are periodic splines under tension and functions of the polygonal arclength parameter.

ACCESS CARDS *FORTRAN,S=ULIB,N=KURVP
 *COSY

These cards access both routines of the package, KURVP.

SPACE REQUIRED Total space for both routines is $627_8 = 407_{10}$

ENTRY POINTS KURVP1, KURVP2

SPECIAL CONDITIONS None

COMMON BLOCKS None

I/O None

PRECISION Single

**REQUIRED ULIB
ROUTINES** None

SPECIALIST Russell K. Rew, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Originally written by A. K. Cline, March, 1972

ALGORITHM

A tridiagonal system (with corner elements) is formed and solved for the vectors XP and YP proportional to the curvature of the curve at the data points.

Curve values are mapped in a two stage process. First the value $ST=T*S$ of polygonal arclength is placed in an interval (S_{i-1}, S_i) where S_i is the polygonal arclength from the first data point to the i^{th} point. Then spline under tension interpolation is done in the two coordinate directions as functions of ST.

ACCURACY

Nearly full precision for SIGMA of about one in magnitude. For $ABS(SIGMA)$ less than .001 accuracy may seriously degenerate due to SINH approximation.

PORTABILITY

Fully portable

**REQUIRED RESIDENT
ROUTINES**

EXP, SQRT

SUBROUTINE KURVP1 (N,X,Y,XP,YP,TEMP,S,SIGMA)

DIMENSION OF ARGUMENTS X(N),Y(N),XP(N),YP(N),TEMP(2*N)

LATEST REVISION October 22, 1973

USAGE CALL KURVP1 (N,X,Y,XP,YP,TEMP,S,SIGMA)

ARGUMENTS

On Input

N

Is the number of points to be interpolated (N.GE.2).

X

Is an array containing the N X-coordinates of the points.

Y

Is an array containing the N Y-coordinates of the points.

XP and YP

Are arrays of length at least N.

TEMP

Is an array of length at least 2*N which is used for scratch storage.

SIGMA

Contains the tension factor. This is a non-zero quantity (whose sign is ignored) which indicates the curviness desired. If ABS(SIGMA) is very large (e.g., 50.) the resulting curve is very nearly a polygon. A standard value for SIGMA is approximately 1. in absolute value.

On Output

XP and YP

Contain information about the curvature of the curve at the given nodes.

S

Contains the polygonal arclength of the curve.

N, X, Y and SIGMA are unaltered.

SPACE REQUIRED

$431_8 = 281_{10}$ locations.

TIMING

Proportional to N. The approximate running time for N=50 on the NCAR 7600 is 1.3 milliseconds.

SUBROUTINE KURVP2 (T,XS,YS,N,X,Y,XP,YP,S,SIGMA)

DIMENSION OF ARGUMENTS X(N),Y(N),XP(N),YP(N)

LATEST REVISION October 22, 1973

USAGE CALL KURVP2 (T,XS,YS,N,X,Y,XP,YP,S,SIGMA)

ARGUMENTS

On Input

T

Contains a real value of absolute value less than or equal to 1. to be mapped to a point on the curve. The sign of T is ignored and the interval (0.,1.) is mapped onto the entire closed curve. If T is negative this indicates that the subroutine has been called previously (with all other input variables unaltered) and that this value of T exceeds the previous value in absolute value. With such information the subroutine is able to map the point much more rapidly. Thus if the user seeks to map a sequence of points onto the same curve, efficiency is gained by ordering the values increasing in magnitude and setting the signs of all but the first, negative.

N

Contains the number of points which were interpolated to determine the curve.

X and Y

Are arrays containing the X- and Y-coordinates of the interpolated points.

XP and YP

Are the arrays output from KURVP1 containing curvature information.

S

Contains the polygonal arclength of the curve.

SIGMA

Contains the tension factor (its sign is ignored).

The parameters N, X, Y, XP, YP, S and SIGMA should be input unaltered from the output of KURVPL.

On Output

XS and YS

Contain the X- and Y-coordinates of the image point on the curve.

T, N, X, Y, XP, YP, S and SIGMA are unaltered.

SPACE REQUIRED

$176_8 = 126_{10}$ locations

TIMING

In efficient mode ($T < 0.$) running time per call on the NCAR CDC 7600 is approximately 55 microseconds.

PACKAGE QURV**LATEST REVISION**

October 1973

PURPOSE

To perform the mapping of points in the interval (0.,1.) onto a curve in space, using splines under tension passing through a sequence of triples $(X(1),Y(1),Z(1)), \dots, (X(N),Y(N),Z(N))$. The slopes at the two ends of the curve may be specified or omitted.

Subroutine QURV1

QURV1 determines the parameters necessary to compute a spline under tension passing through a sequence of triples $(X(1),Y(1),Z(1)), \dots, (X(N),Y(N),Z(N))$ in space. The slopes at the two ends of the curve may be specified or omitted. For actual computation of points on the curve it is necessary to call the subroutine QURV2.

PURPOSE*(continued)*

Subroutine QURV2

QURV2 performs the mapping of points in the interval (0.,1.) onto a curve in space. The subroutine QURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation all of whose components are splines under tension and functions of the polygonal arclength parameter.

ACCESS CARDS

*FORTRAN, =ULIB,N=QURV

*COSY

These cards access both routines of the package, QURV.

SPACE REQUIREDTotal space for both routines is $1140_8 = 608_{10}$.**ENTRY POINTS**

QURV1, QURV2

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB ROUTINES None

SPECIALIST Russell K. Rew, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Originally written by A.K. Cline, July 1972

ALGORITHM
A tridiagonal system is formed and solved for the vectors XP, YP and ZP, of values proportional to the curvature of the curve at the data points.

Curve values are mapped in a two stage process. First the value $ST = T*S$ of polygonal arclength is placed in an interval (S_{i-1}, S_i) where S_i is the polygonal arclength from the first data point to the i^{th} point. Then spline under tension interpolation is done in the three coordinate directions as functions of ST.

ACCURACY Nearly full precision for SIGMA of about one in magnitude. For ABS(SIGMA) less than .001 accuracy may seriously degenerate due to SINH approximation.

PORTABILITY Fully portable

REQUIRED RESIDENT ROUTINES EXP, SQRT

**SUBROUTINE QURV1 (N,X,Y,Z,SLP1X,SLP1Y,SLP1Z,SLPNX,SLPNY,SLPNZ,XP,YP,
ZP,TEMP,S,SIGMA)**

**DIMENSION OF
ARGUMENTS**

X(N),Y(N),Z(N),XP(N),YP(N),ZP(N),TEMP(N)

LATEST REVISION

October 22, 1973

USAGE

CALL QURV1 (N,X,Y,Z,SLP1X,SLP1Y,SLP1Z,SLPNX,SLPNY,SLPNZ,XP,
YP,ZP,TEMP,S,SIGMA)

ARGUMENTS

On Input

N

Is the number of points to be interpolated (N.GE.2).

X

Is an array containing the X-coordinates of the points.

Y

Is an array containing the Y-coordinates of the points.

Z

Is an array containing the Z-coordinates of the points.

SLP1X, SLP1Y, SLP1Z and SLPNX, SLPNY, SLPNZ

Contain the desired values of the components of a tangent vector to the curve at (X(1),Y(1),Z(1)) and (X(N),Y(N),Z(N)) respectively. The positive sense of the curve is assumed to be that moving from point 1 to point N. If the quantity SIGMA is negative these slopes will be determined internally and the user need only furnish place-holding (dummy) parameters for SLP1X, SLP1Y, SLP1Z, SLPNX, SLPNY and SLPNZ. Such place-holding parameters will be ignored but not destroyed.

On Input*(continued)*

XP, YP and ZP

Are arrays of length at least N.

TEMP

Is an array of length at least N which is used for scratch storage.

SIGMA

Contains the tension factor. This is non-zero and indicates the curviness desired. If ABS(SIGMA) is very large (e.g., 50.) the resulting curve is very nearly a polygonal line. The sign of SIGMA indicates whether slope information has been input or not. If SIGMA is negative, the end-point slopes will be determined internally. A standard value for SIGMA is approximately 1. in absolute value.

On Output

XP, YP and ZP

Contain information about the curvature of the curve at the given nodes.

S

Contains the polygonal arclength of the curve.

N, X, Y, Z, SLP1X, SLP1Y, SLP1Z, SLPNX, SLPNY, SLPNZ and SIGMA are unaltered.

SPACE REQUIRED701₈ = 449₁₀ locations**TIMING**

Proportional to N. For N = 50 the running time on the NCAR CDC 7600 is approximately 2.3 milliseconds.

SUBROUTINE QURV2 (T,XS,YS,ZS,N,X,Y,Z,XP,YP,ZP,S,SIGMA)

**DIMENSION OF
ARGUMENTS**

X(N),Y(N),Z(N),XP(N),YP(N),ZP(N)

October 22, 1973

CALL QURV2 (T,XS,YS,ZS,N,X,Y,Z,XP,YP,ZP,S,SIGMA)

On Input

T

Contains a real value of absolute value less than or equal to 1. to be mapped to a point on the curve. The sign of T is ignored and the interval (0.,1.) is mapped onto the entire curve. If T is negative this indicates that the subroutine has been called previously (with all other input variables unaltered) and that this value of T exceeds the previous value in absolute value. With such information the subroutine is able to map the point much more rapidly. Thus if the user seeks to map a sequence of points onto the same curve, efficiency is gained by ordering the values increasing in magnitude and setting the signs of all but the first, negative.

N

Contains the number of points which were interpolated to determine the curve.

X, Y and Z

Are arrays containing the X-, Y- and Z-coordinates of the interpolated points.

XP, YP and ZP

Are the arrays output from QURV2 containing the curvature information.

On Input
(continued)

S
Contains the polygonal arclength of the curve.

SIGMA
Contains the tension factor (its sign is ignored).

The parameters N, X, Y, Z, XP, YP, ZP, S and SIGMA should be input unaltered from the output of QURV1.

On Output

XS, YS and ZS
Contain the X-, Y- and Z-coordinates of the image point on the curve.

T, N, X, Y, Z, XP, YP, ZP, S and SIGMA are unaltered.

SPACE REQUIRED

$237_8 = 159_{10}$ locations

TIMING

In efficient mode ($T < 0.$) the running time per call on the NCAR CDC 7600 is approximately 62 microseconds.

PACKAGE SPLPAK**PURPOSE**

This package contains routines for fitting (least squares) a multidimensional cubic spline to arbitrarily located data. It also contains routines for evaluating this spline (or its partial derivatives) at any point.

Coefficient calculation is performed in subroutines SPLCC or SPLCW and evaluation is performed by functions SPLFE or SPLDE.

ACCESS CODES

*FORTRAN,S=ULIB,N=SPLPAK
*COSY

ENTRY POINTS

SPLCC, SPLCW, SPLFE, SPLDE, BASCMP, SUPRLS, SUPRSL

SPACE REQUIRED

3200_e (including SUPRLS package)

SUBROUTINE SPLCC (NDIM,XDATA,L1XDAT,YDATA,NDATA,XMIN,XMAX,NODES,
XTRAP,COEF,NCF,WORK,NWRK,IERROR)

DIMENSION OF ARGUMENTS XDATA(L1XDAT,NDATA),YDATA(NDATA),XMIN(NDIM),XMAX(NDIM),
NODES(NDIM),COEF(NCF),WORK(NWRK)

PURPOSE N-dimensional cubic spline coefficient calculation by least squares.

The usage and arguments of this routine are identical to those for SPLCW except the omission of the array of weights, WDATA. See SPLCW for a complete description.

SUBROUTINE SPLCW (NDIM,XDATA,L1XDAT,YDATA,WDATA,NDATA,XMIN,XMAX,
NODES,XTRAP,COEF,NCF,WORK,NWRK,IERROR)

DIMENSION OF ARGUMENTS XDATA(L1XDAT,NDATA),YDATA(NDATA),WDATA(NDATA),XMIN(NDIM),
XMAX(NDIM),NODES(NDIM),COEF(NCF),WORK(NWRK)

LATEST REVISION November 13, 1973

PURPOSE N-dimensional cubic spline coefficient calculation by
weighted least squares on arbitrarily located data.

A grid of evenly spaced nodes in NDIM space is defined by the arguments XMIN, XMAX, and NODES. A linear basis for the class of natural splines on these nodes is formed, and a set of corresponding coefficients is computed in the array COEF. These coefficients are chosen to minimize the weighted sum of squared errors between the spline and the arbitrarily located data values described by the arguments XDATA, YDATA, and NDATA. The smoothness of the spline in data sparse areas is controlled by the argument XTRAP.

NOTE

In order to understand the arguments of this routine, one should realize that the node grid need not bear any particular relation to the data points. In the theory of exact fit interpolatory splines the nodes would in fact be data locations, but in this case they serve only to define the class of splines from which the approximating function is chosen. This node grid is a rectangular arrangement of points in NDIM space, with the restriction that along any coordinate direction the nodes are equally spaced. The class of natural splines on this grid of nodes (NDIM-cubic splines whose 2nd derivatives normal to the boundaries are 0) has as many degrees of freedom as the grid has nodes. Thus the smoothness or flexibility of the splines is determined by the choice of the node grid.

USAGE

Call SPLCW (NDIM,XDATA,LLXDAT,YDATA,WDATA,NDATA,XMIN,XMAX,
NODES,XTRAP,COEF,NCF,WORK,NWRK,IERROR)

The spline (or its derivatives) may then be evaluated by using function SPLFE (or SPLDE).

ARGUMENTS**On Input**

NDIM

The dimensionality of the problem. The spline is a function of NDIM variables or coordinates and thus a point in the independent variable space is an NDIM vector. NDIM must be in the range $1 \leq \text{NDIM} \leq 4$.

XDATA

A collection of locations for the data values, i.e. points from the independent variable space. This collection is a 2-dimensional array whose 1st dimension indexes the NDIM coordinates of a given point and whose 2nd dimension labels the data point. For example, the data point with label IDATA is located at the point (XDATA(1, IDATA), ..., XDATA(NDIM, IDATA)) where the elements of this vector are the values of the NDIM coordinates. The location, number, and ordering of the data points is arbitrary.

The dimension is assumed to be XDATA(L1XDAT, NDATA).

L1XDAT

The length of the 1st dimension of XDATA in the calling program. L1XDAT must be \geq NDIM.

Note: For 1 dimensional problems L1XDAT is usually 1.

YDATA

A collection of data values corresponding to the points in XDATA. YDATA(IDATA) is the data value associated with the point (XDATA(1, IDATA), ..., XDATA(NDIM, IDATA)) in the independent variable space. The spline whose coefficients are computed by this routine approximates these data values in the least squares sense.

The dimension is assumed to be YDATA(NDATA).

WDATA

A collection of weights. WDATA(IDATA) is a weight associated with the data point labelled IDATA. It should be non-negative, but may be of any magnitude. The weights have the effect of forcing greater or lesser accuracy at a given point as follows--this routine chooses coefficients to minimize the sum over all data points of the quantity $(WDATA(IDATA) * (YDATA(IDATA) - \text{spline value at XDATA(IDATA)}))^2$. Thus, if the reliability of a data point is known to be low, the corresponding weight may be made small (relative to the other weights) so that the sum over all data points is affected less by discrepancies at the unreliable point. Data points with zero weight are completely ignored.

Note: If WDATA(1) is < 0 , the other elements of WDATA are not referenced, and all weights are assumed to be unity.

The dimension is assumed to be WDATA(NDATA) unless WDATA(1) < 0 in which case the dimension is assumed to be 1.

On Input
(continued)

NDATA

The number of data points mentioned in the above arguments.

XMIN

A vector describing the lower extreme corner of the node grid. A set of evenly spaced nodes is formed along each coordinate axis and XMIN(IDIM) is the location of the first node along the IDIM axis.

The dimension is assumed to be XMIN(NDIM).

XMAX

A vector describing the upper extreme corner of the node grid. A set of evenly spaced nodes is formed along each coordinate axis and XMAX(IDIM) is the location of the last node along the IDIM axis.

The dimension is assumed to be XMAX(NDIM).

NODES

A vector of integers describing the number of nodes along each axis. NODES(IDIM) is the number of nodes (counting endpoints) along the IDIM axis and determines the flexibility of the spline in that coordinate direction. NODES(IDIM) must be ≥ 4 but may be as large as the arrays COEF and WORK allow.

The dimension is assumed to be NODES(NDIM).

Note: The node grid is completely defined by the arguments XMIN, XMAX, and NODES. The spacing of this grid in the IDIM coordinate direction is $DX(IDIM) = (XMAX(IDIM) - XMIN(IDIM)) / (NODES(IDIM) - 1)$. A node in this grid may be indexed by an NDIM vector of integers (IN(1), ..., IN(NDIM)) where $1 \leq IN(IDIM) \leq NODES(IDIM)$. The location of such a node may be represented by an NDIM vector (X(1), ..., X(NDIM)) where $X(IDIM) = XMIN(IDIM) + (IN(IDIM) - 1) * DX(IDIM)$.

XTRAP

A parameter to control extrapolation to data sparse areas. The region described by XMIN and XMAX is divided into rectangles, the number of which is determined by NODES, and any rectangle containing a disproportionately small number of data points is considered to be data sparse (rectangle is used here to mean NDIM-dimensional rectangle). If XTRAP is nonzero the least squares problem is augmented with derivative constraints in the data sparse areas to prevent the matrix from becoming poorly conditioned. XTRAP serves as a weight for these constraints, and thus may be used to control smoothness in data sparse areas. The experience of the author indicates that unity is a good first guess for this parameter, and that its size is not very critical.

Note: If XTRAP is 0 substantial portions of the routine will be skipped, but a singular matrix can result if large portions of the region are without data.

NCF

The length of the array COEF in the calling program. If NCF is $< \text{NODES}(1)*, \dots, *\text{NODES}(\text{NDIM})$ a fatal error is diagnosed.

WORK

A workspace array for solving the least squares matrix generated by this routine. Its required size is a function of the total number of nodes in the node grid. This total, $\text{NCOL} = \text{NODES}(1)*, \dots, *\text{NODES}(\text{NDIM})$, is also the number of columns in the least squares matrix. The length of the array WORK must equal or exceed $\text{NCOL}*(\text{NCOL}+8)/2$ (OR $\text{NCOL}*(\text{NCOL}+6)/2$ if XTRAP is 0).

The dimension is assumed to be WORK(NWRK).

NWRK

The length of the array WORK in the calling program. If $\text{NCOL} = \text{NODES}(1)*, \dots, *\text{NODES}(\text{NDIM})$ is the total number of nodes, then a fatal error is diagnosed if NWRK is less than $\text{NCOL}*(\text{NCOL}+8)/2$ (OR $\text{NCOL}*(\text{NCOL}+6)/2$ if XTRAP is 0).

On Output**COEF**

The array of coefficients computed by this routine. Each coefficient corresponds to a particular basis function which in turn corresponds to a node in the node grid. This correspondence between the node grid and the array COEF is as if COEF were an NDIM-dimensional Fortran array with dimensions NODES(1),...,NODES(NDIM), i.e., to store the array linearly, the leftmost indices are incremented most frequently. Hence the length of the COEF array must equal or exceed the total number of nodes, which is $\text{NODES}(1)*\dots*\text{NODES}(\text{NDIM})$. The computed array COEF may be used with function SPLFE (or SPLDE) to evaluate the spline (or its derivatives) at an arbitrary point in NDIM space.

The dimension is assumed to be COEF(NCF).

WORK

The workspace containing intermediate calculations. It need not be saved.

IERROR

An error flag with the following meanings--

0	No error
101	NDIM is < 1 or is > 4
102	NODES(IDIM) is < 4 for some IDIM
103	XMIN(IDIM) = XMAX(IDIM) for some IDIM
104	NCF (size of COEF) is < $\text{NODES}(1)*\dots*\text{NODES}(\text{NDIM})$
105	NDATA is < 1
106	NWRK (size of WORK) is too small--see comments

COMMON BLOCKS

SPLCOM

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES****SUPRLS**

A least squares matrix package. Since this package has not yet been standardized, it is not on the ULIB. Therefore it is currently part of this package and need not be loaded separately.

SPECIALIST

Dave Fulker, NCAR, Boulder, Colorado, 80303.

LANGUAGE

FORTRAN

HISTORY

Developed 1972-1973. Standardized July 13, 1973. Revised August 1973 and October 1973.

ALGORITHM

An overdetermined system of linear equations is formed-- one equation for each data point plus equations for derivative constraints. This system is solved using the ULIB package SUPRLS which see for description.

ACCURACY

If there is exactly one data point in the near vicinity of each node and no extra data, the resulting spline will agree with the data values to 12 or 13 digits. However, if the problem is overdetermined or the sparse data option is utilized, the accuracy is hard to predict. Basically, smooth functions require fewer nodes than rough ones for the same accuracy.

TIMING

The executive time is roughly proportional to $N_{DATA} * N_{COF}^{**2}$ where $N_{COF} = N_{NODES}(1) * \dots * N_{NODES}(N_{DIM})$. On the CDC 7600 the proportionality factor is about .002 milliseconds although it is larger when N_{COF} is very small. When $XTRAP$ is not zero the execution time will be substantially increased if data sparse areas are encountered.

PORTABILITY

A Fortran version of ULIBER would be needed.

**REQUIRED RESIDENT
ROUTINES**

ULIBER (an error message printing routine)

FUNCTION SPLFE (NDIM,X,COEF,XMIN,XMAX,NODES,IERROR)

DIMENSION OF ARGUMENTS X(NDIM),NDERIV(NDIM),COEF(NODES(1),*...*,NODES(NDIM)),
XMIN(NDIM),XMAX(NDIM),NODES(NDIM)

NOTE COEF, XMIN, XMAX, and NODES must be exactly retained from the call to SPLCC (or SPLCW).

PURPOSE N-dimensional cubic spline function evaluation.

Except for lack of derivative capability, this routine is identical to SPLDE in usage. The argument list is also identical except for the omission of NDERIV. See SPLDE for a complete description.

FUNCTION SPLDE (NDIM,X,NDERIV,COEF,XMIN,XMAX,NODES,IERROR)

DIMENSION OF ARGUMENTS X(NDIM),NDERIV(NDIM),COEF(NODES(1),*...*,NODES(NDIM)), XMIN(NDIM),XMAX(NDIM),NODES(NDIM)

NOTE COEF, XMIN, XMAX, and NODES must be exactly retained from the call to SPLCC (or SPLCW).

LATEST REVISION November 13, 1973

PURPOSE N-dimensional cubic spline derivative evaluation.

A grid of evenly spaced nodes in NDIM space is defined by the arguments XMIN, XMAX, and NODES. A linear basis for the class of natural splines on these nodes is formed, and to each basis function corresponds a coefficient in the array COEF (computed in SPLCC or SPLCW). Using NDERIV to indicate the appropriate partial derivatives, each basis function is evaluated at the point X in NDIM space. These values are then multiplied by the corresponding coefficient and summed to form the function result.

USAGE First, the array COEF must have been prepared by using subroutine SPLCC (or SPLCW). Then to evaluate partial derivatives of the spline set $Y = \text{SPLDE}(\text{NDIM}, X, \text{NDERIV}, \text{COEF}, \text{XMIN}, \text{XMAX}, \text{NODES}, \text{IERROR})$.

ARGUMENTS

On Input

NDIM

The dimensionality of the problem. The spline is a function of NDIM variables or coordinates and thus a point in the independent variable space is an NDIM vector. NDIM must be in the range $1 \leq \text{NDIM} \leq 4$.

X

An NDIM vector describing the point in the independent variable space at which the spline is to be evaluated.

The dimension is assumed to be $X(\text{NDIM})$.

NDERIV

An NDIM vector of integers specifying the partial derivative to be evaluated. The order of the derivative along the IDIM axis is $\text{NDERIV}(\text{IDIM})$. These integers must be in the range $0 \leq \text{NDERIV}(\text{IDIM}) \leq 2$.

COEF

The array of coefficients which determine the spline. Each coefficient corresponds to a particular basis function which in turn corresponds to a node in the node grid. This correspondence between the node grid and the array COEF is as if COEF were an NDIM-dimensional Fortran array with dimensions $\text{NODES}(1), \dots, \text{NODES}(\text{NDIM})$, i.e., to store the array linearly, the leftmost indices are incremented most frequently. COEF may be computed by using routines SPLCC or SPLCW.

The dimension is assumed to be $\text{COEF}(\text{NODES}(1)*\dots*\text{NODES}(\text{NDIM}))$.

XMIN

A vector describing the lower extreme corner of the node grid. A set of evenly spaced nodes is formed along each coordinate axis and $\text{XMIN}(\text{IDIM})$ is the location of the first node along the IDIM axis.

The dimension is assumed to be $\text{XMIN}(\text{NDIM})$.

XMAX

A vector describing the upper extreme corner of the node grid. A set of evenly spaced nodes is formed along each coordinate axis and $\text{XMAX}(\text{IDIM})$ is the location of the last node along the IDIM axis.

On Input
(continued)

The dimension is assumed to be XMAX(NDIM).

NODES

A vector of integers describing the number of nodes along each axis. NODES(IDIM) is the number of nodes (counting endpoints) along the IDIM axis and determines the flexibility of the spline in that coordinate direction. NODES(IDIM) must be ≥ 4 but may be as large as the arrays COEF and WORK allow.

The dimension is assumed to be NODES(NDIM).

Note: The node grid is completely defined by the arguments XMIN, XMAX, and NODES. The spacing of this grid in the IDIM coordinate direction is $DX(IDIM) = (XMAX(IDIM) - XMIN(IDIM)) / (NODES(IDIM) - 1)$. A node in this grid may be indexed by an NDIM vector of integers (IN(1), ..., IN(NDIM)) where $1 \leq IN(IDIM) \leq NODES(IDIM)$. The location of such a node may be represented by an NDIM vector (X(1), ..., X(NDIM)) where $X(IDIM) = XMIN(IDIM) + (IN(IDIM) - 1) * DX(IDIM)$.

On Output**SPLDE**

The function value returned is the partial derivative (indicated by NDERIV) of the spline evaluated at X.

IERROR

An error flag with the following meanings:

- 0 No error
- 101 NDIM is < 1 or is > 4
- 102 NODES(IDIM) is < 4 for some IDIM
- 103 XMIN(IDIM) = XMAX(IDIM) for some IDIM
- 104 NDERIV(IDIM) is < 0 or is > 2 for some IDIM

COMMON BLOCKS

SPLCOM

I/O

None

PRECISION

Single

REQUIRED ULIB ROUTINES ULIBER (on system)

SPECIALIST Dave Fulker, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Developed 1972-1973. Standardized July 13, 1973. Revised October 1973.

ALGORITHM The multi-dimensional basis functions are formed as tensor products of 1-dimensional basis functions. These are evaluated at X, multiplied by the appropriate coefficient, and summed to form the result.

ACCURACY Essentially machine precision.

TIMING Approximately $.2 \times 3^{**}(\text{NDIM}-1)$ milliseconds on the CDC 7600.

PORTABILITY A Fortran version of ULIBER would be needed.

REQUIRED RESIDENT ROUTINES ULIBER (an error message printing routine).

PACKAGE SURF**LATEST REVISION** October 1973**PURPOSE**

To perform two-dimensional interpolation on a rectangular grid of functional values using a bi-spline under tension. The X- and Y-derivatives around the boundary and the XY derivatives at the four corners may be specified or omitted.

Subroutine SURF1

SURF1 determines the parameters necessary to compute an interpolatory surface passing through a rectangular grid of functional values. The surface determined can be represented as the tensor product of splines under tension. The X- and Y-derivatives around the boundary and the XY derivatives at the four corners may be specified or omitted. For actual mapping of points onto the surface it is necessary to call the function SURF2.

Function SURF2

SURF2 interpolates a surface at a given coordinate pair using a bi-spline under tension. The subroutine SURF1 should be called earlier to determine certain necessary parameters.

ACCESS CARDS

*FORTRAN,S=ULIB,N=SURF
*COSY

These cards access both the routine SURF1 and the function SURF2 of the package, SURF.

SPACE REQUIRED

Total space for both the routine and the function is
 $3365_8 = 1782_{10}$

ENTRY POINTS

SURF1, SURF2

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Russell K. Rew, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Originally written by A.K. Cline, September 1973.

ALGORITHM

After determining boundary derivatives (if not user specified) a tridiagonal linear system based upon the Y-directional grid is formed and values proportional to f_{yy} are determined along each X-grid line. (Also f_{xyy} is determined along X(1) and X(M) grid lines.) Then another tridiagonal system based upon the X-directional grid is formed and values proportional to f_{xx} and f_{xxyy} are determined along each Y-grid line. As a result values proportional to f_{xx} , f_{yy} , and f_{xxyy} are determined at every grid point.

Except for points outside the grid, a grid rectangle $(X_{i-1}, X_i) \times (Y_{j-1}, Y_j)$ is located containing (XX,YY). Four spline under tension interpolations are performed in the y-direction first to determine $f(X_{i-1}, YY)$, $f(X_i, YY)$, $f_{xx}(X_{i-1}, YY)$, and $f_{xx}(X_i, YY)$. Finally an X-directional spline under tension interpolation is performed using these four values to determine $f(XX, YY)$.

ACCURACY

Nearly full precision for SIGMA of about one in magnitude. For ABS(SIGMA) less than .001 accuracy may seriously degenerate due to SINH approximation.

PORTABILITY

Fully portable.

**REQUIRED RESIDENT
ROUTINES**

EXP

**SUBROUTINE SURF1 (M,N,X,Y,Z,IZ,ZX1,ZXM,ZY1,ZYN,ZXY11,ZXYM1,ZXY1N,
ZXYMN,ZP,TEMP,SIGMA)**

DIMENSION OF ARGUMENTS X(M),Y(N),Z(IZ,N),ZX1(N),ZXM(N),ZY1(M),ZYN(M),ZP(3*M*N),
TEMP(2*M+N)

LATEST REVISION October 22, 1973

USAGE CALL SURF1 (M,N,X,Y,Z,IZ,ZX1,ZXM,ZY1,ZYN,ZXY11,ZXYM1,ZXY1N,
ZXYMN,ZP,TEMP,SIGMA)

ARGUMENTS

On Input

M

Is the number of grid lines in the X-direction, i.e., lines parallel to the Y axis (M.GE.2).

N

Is the number of grid lines in the Y-direction, i.e., lines parallel to the X-axis (N.GE.2).

X

Is an array containing the X-coordinates of the M grid lines in the X-direction.

Y

Is an array containing the Y-coordinates of the M grid lines in the Y-direction.

Z

Is an array containing the M*N functional values at the grid points, i.e., Z(I,J) contains the functional value at (X(I),Y(J)) I = 1,...,M J = 1,...,N.

On Input
(continued)

IZ

Is the FORTRAN row dimension of the matrix Z used in the calling program (IZ.GE.M).

ZX1

Is an array of X-derivatives of the function along the X(1) grid line, i.e., ZX1(H) contains the X-partial derivative at the point (X(1),Y(J)) J = 1,...,N. (This parameter is ignored if SIGMA.LT.0.).

ZXM

Is an array of X-derivatives of the function along the X(M) grid line, i.e., ZXM(J) contains the X-partial derivative at the point (X(M),Y(J)) J = 1,...,N. (This parameter is ignored if SIGMA.LT.0.).

ZY1

Is an array of Y-derivatives of the function along the Y(1) grid line, i.e., ZY1(I) contains the Y-partial derivative at the point (X(I),Y(1)) I = 1,...,M. (This parameter is ignored if SIGMA.LT.0.).

ZYN

Is an array of Y-derivatives of the function along the Y(N) grid line, i.e., ZYN(I) contains the Y-partial derivative at the point (X(I),Y(N)) I = 1,...,M. (This parameter is ignored if SIGMA.LT.0.).

ZXY11, ZXYM1, ZXY1N and ZXYMN

Are the XY-derivatives of the function at the four corners of the grid, i.e., ZXY11 (ZXYM1,ZXY1N,ZXYMN) contains the X-Y second partial derivative at the point (X(1),Y(1)) ((X(M),Y(1)),(X(1),Y(N)),(X(M),Y(N)), respectively). (These parameters are ignored if SIGMA.LT.0.).

ZP

Is an array of at least 3*M*N locations.

TEMP

Is an array of at least 2*M+N locations used for internal temporary storage.

On Input
(continued)

SIGMA

Contains the tension factor. This is non-zero and indicates the curviness desired. If ABS(SIGMA) is nearly zero (e.g., .001) the resulting surface is a bi-cubic spline. If ABS(SIGMA) is large (e.g., 50.) the resulting surface is nearly bi-linear. If SIGMA is negative, values for the boundary and corner partial derivatives will be determined internally and only place-holding (dummy) parameters need be given in the call for arguments ZX1, ZXM, ZY1, ZYN, ZXY11, ZXYM1, ZXY1N and ZXYMN. A standard value for SIGMA is approximately 1. in absolute value.

On Output

ZP

Contains values proportional to XX-, YY- and XXYY-partial derivatives at the grid points to be used for the actual interpolation.

M, N, X, Y, Z, IZ, ZX1, ZXM, ZY1, ZYN, ZXY11, ZXYM1, ZXY1N, ZXYMN and SIGMA are unaltered.

SPACE REQUIRED

2705₈ = 1477₁₀ locations

TIMING

Proportional to M*N. For M = 50, N = 40 the running time on the NCAR CDC 7600 is approximately 21 milliseconds.

FUNCTION SURF2 (XX,YY,M,N,X,Y,Z,IZ,ZP,SIGMA)**DIMENSION OF ARGUMENTS**

X(M),Y(N),Z(IZ,N),ZP(3*M*N)

LATEST REVISION

October 22, 1973

USAGE

ZXY = SURF2 (XX,YY,M,N,X,Y,Z,IZ,ZP,SIGMA)

ARGUMENTS**On Input**

XX and YY

Contain the X- and Y-coordinates of the point to be mapped onto the interpolating surface.

M and N

Contain the number of grid lines in the X- and Y-directions respectively, of the rectangular grid which determined the surface.

X and Y

Are arrays containing X- and Y-grid values respectively, each in increasing order.

Z

Is a matrix containing the M*N functional values corresponding to the grid values (i.e., Z(I,J) is the surface value at the point (X(I),Y(J)) I = 1,...,M J = 1,...,N).

IZ

Contains the FORTRAN row dimension of the array Z declared in the calling program.

On Input
(continued)

ZP

Is an array of $3 \cdot M \cdot N$ locations stored with the various surface derivative values determined by SURF1.

SIGMA

Contains the tension factor (its sign is ignored).

The parameters M, N, X, Y, Z, IZ, ZP and SIGMA should be input unaltered from the output of SURF1.

On Output

SURF2

Contains the interpolated surface value.

None of the input parameters are altered.

SPACE REQUIRED

$460_8 = 304_{10}$ locations

TIMING

For a large ($>M \cdot N$) number of calls with XX and YY values in non-decreasing orders the running time per call is approximately 88 microseconds.

PACKAGE TRIANGLE**LATEST REVISION**

May 1974

PURPOSE

To perform interpolation over a two dimensional set of points. Given N points $(X(I), Y(I))$, $I = 1, 2, \dots, N$ and N corresponding function values, $Z(I)$, this package sets up a grid of triangles with vertices at the given points. To interpolate at the point (XX, YY) , linear interpolation is performed within the triangle containing (XX, YY) . If the point lies outside the triangulation extrapolation is carried out in such a way as to produce a function which is continuous over the whole plane.

The package contains nine routines, of which only the first two need be referenced by the user.

Subroutine TRIANG

This produces a triangulation of the given data points.

Function TRINTR

Given the triangulation of a two dimensional set of points produced by TRIANG, this function performs piecewise linear interpolation.

PURPOSE
(continued)

The remaining routines are internal to the interpolation package and need not be referenced by the user. They are described in detail so that the triangulation procedure may be used in other contexts.

Subroutine CONHUL

This is called by TRIANG to search for the convex hull of the given point set, i.e., the smallest convex region containing the point set.

Subroutine TMESH

This is called by TRIANG to set up a triangulation within the convex hull found by CONHUL.

Subroutine TMESHI

TMESHI is called by TRIANG to improve the initial triangulation produced by TMESH.

Subroutine SORT1 (with second entry point SORT2)

This routines performs a sort of a given set of records according to criteria provided by the user. The result is in the form of a linked chain. The routine is used by CONHUL to perform a major sort on increasing $X(I)$ coupled with a minor sort on decreasing $Y(I)$.

Subroutine PVEC

PVEC is called by CONHUL to convert the linked chain produced by SORT1 to a permutation vector .

Subroutine STEST

This routine is used to test an interior line of the triangulation for possible improvement. Each interior line is a diagonal of a quadrilateral. The effect of substituting this line by the other diagonal is examined and the line which

produces the largest minimum interior angle in the two adjacent triangles is accepted. The routine is called by TMESH and TMESHI.

Function LTEST

This function evaluates the cross product of two vectors (SX, SY) and (DX, DY) . If the angle between them is less than π , the value, + 1, is returned. If the vectors are parallel and (DX, DY) is a multiple of (SX, SY) between 0. and 1., the value 0 is returned. Otherwise, LTEST is set to -1. LTEST is called by TMESH in a procedure for identifying the triangle within which a given point lies.

ACCESS CARDS

*FORTRAN,S=ULIB,N=TRIANGLE
*COSY

These cards access the nine routines TRIANG, CONHUL, TMESH, TMESHI, SORT1, PVEC, STEST, LTEST, TRINTR.

SPACE REQUIRED

Total space for all nine routines is $4606_8 = 2438_{10}$.

ENTRY POINTS

TRIANG, CONHUL, TMESH, TMESHI, SORT1, SORT2, PVEC, STEST
LTEST, TRINTR

SPECIAL CONDITIONS

This interpolation procedure works best for a reasonably uniform density of points. Less satisfactory results will be obtained if the point density fluctuates widely.

2.TRIANGLE.4

COMMON BLOCKS

TRIAN1, TRIAN2

TRIAN1 is of length $12_8 = 10_{10}$ and is common to TRIANG, CONHUL, TMESH, TMESHI, STEEST and TRINTR.

TRIAN2 is of length 4 and is common to TMESHI and STEEST.

I/O

None

PRECISION

Single

REQUIRED ULIB ROUTINES

None

SPECIALIST

Cicely Ridley, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

The routines CONHUL, TMESH, TMESHI, LTEST, STEEST, SORT1, PVEC are standardized versions of routines described by C.L. Lawson in his fascinating study: Generation of a triangular grid with application to contour plotting. Section 914, Technical Memorandum No. 299, 1972, Jet Propulsion Laboratory, California Institute of Technology. The routines TRIANG and TRINTR are standardized versions of routines written by A. K. Cline.

ACCURACY

A test was carried out using a uniform random distribution of 100 points within a unit square. The corresponding values of Z were taken to lie on the plane, $Z = 1. + X + 4.$ For points within the convex hull, interpolation to near machine accuracy resulted.

PORTABILITY

Machine dependent constants are defined by data statements in TMESH, STEST and LTEST.

These are clearly commented.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, SQRT

SUBROUTINE TRIANG (X,Y,N,ITRI,ISUP,IUSED,IER)

DIMENSION OF ARGUMENTS X(N),Y(N),ITRI(ISUP)

PURPOSE To determine a triangulation of a given two dimensional set of points.

LATEST REVISION May 1974

USAGE CALL TRIANG (X,Y,N,ITRI,ISUP,IUSED,IER)

ARGUMENTS

On Input X
Array containing X coordinates of the N given points. The dimension of X in the calling program must be at least N.

Y
Array containing Y coordinates of the N given points. The dimension of Y in the calling program must be at least N.

N
Number of given (X,Y) data points.

ISUP
Dimension of ITRI in calling program. ISUP should be at least $(22*N-43)$.

On Output**I TRI**

Vector in which details of triangulation are returned.

IUSED

Number of elements of I TRI used to describe triangulation. This depends upon how many points lie on the boundary. The upper limit is $(22*N-43)$ when the convex hull contains only three points.

I ER

Error parameter.

= 0 Triangulation successfully completed.

Non-fatal errors.

= 1 Duplicate point removed.

= 2 Point lies on boundary but was not included in convex hull.

= 3 Iterature improvement terminated on counter.

Fatal errors.

= 32 Require at least 4 distinct points.

= 33 All points are in a line.

= 34 All $X(I) = XMIN$

= 35 No interior points.

= 36 Interior point not included in any triangle.

= 37 Data lists inconsistent.

COMMON BLOCKS

TRIANG of length $12_8 = 10_{10}$

ALGORITHM

This routine calls CONHUL to determine the convex hull of the given set of data points. TMESH is then called to set up an initial triangulation within the hull. Finally, TMESHI carries out iterative improvement of the triangulation.

2.TRIANGLE.8

SPACE REQUIRED

$157_8 = 111_{10}$

TIMING

The time required to produce a triangulation for 100 points having a uniform random distribution over a unit square was 296 msec on the CDC 7600.

PORTABILITY

This routine is portable.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

FUNCTION TRINTR (XX,YY,X,Y,Z,N,ITRI)**DIMENSION OF ARGUMENTS**

X(N),Y(N),Z(N),ITRI(USED). Where IUSED is returned by the routine TRIANG (see above).

LATEST REVISION

May 1974

PURPOSE

Given a two dimensional set of data points (X,Y) and corresponding function values, Z, this function subroutine returns the result of piecewise triangular surface interpolation at any desired point in the (X,Y) plane. It makes use of a triangulation previously produced by the subroutine TRIANG. For points outside the triangulation, extrapolation is performed to produce a function continuous over the whole plane.

USAGE

F = TRINTR (XX,YY,X,Y,Z,ITRI)

F = interpolated value at the point (XX,YY).

ARGUMENTS**On Input**

XX

X-coordinate of the point to be interpolated.

YY

Y-coordinate of the point to be interpolated.

X

Array of x coordinates of given data points. Dimension of X in calling program must be at least N, where N is total number of points.

On Input
(continued)

Y

Array of corresponding Y-coordinates. Dimension of Y in calling program must be at least N.

Z

Array such that Z(I) is the given function value associated with data point (X(I),Y(I)). Dimension of Z in calling program must be at least N.

N

Number of data points.

ITRI

Array containing triangulation information produced by TRIANG.

On Output

TRINTR

Contains the interpolated value at (XX,YY).

COMMON BLOCKS

TRIAN1 of length $12_8 = 10_{10}$

ALGORITHM

The triangulation is scanned to identify the triangle containing the point at which interpolation is required. Linear interpolation is then performed within that triangle. A point lying outside the triangulation is given a value which makes the function continuous over the whole face.

SPACE REQUIRED

$454_8 = 300_{10}$

TIMING

Time to perform 100 interpolations on a triangulation involving 100 points was 203 msec.

PORTABILITY

This routine is portable.

REQUIRED RESIDENT
ROUTINES

None

SUBROUTINE CONHUL (X,Y,NPTS,IP,IP2,IER)

DIMENSION OF ARGUMENTS

X(NPTS),Y(NPTS),IP(NPTS+1),IP2(NPTS+1)

LATEST REVISION

May 1974

PURPOSE

To find the convex hull of a two dimensional set of points (X(I),Y(I)), I = 1,2,...,NPTS.

USAGE

CALL CONHUL (X,Y,NPTS,IP,IP2,IER)

ARGUMENTS

On Input

X

Array containing X coordinates of the NPTS given points. Dimension of X in calling program must be at least NPTS.

Y

Array containing Y coordinates of the NPTS given points. Dimension of Y in calling program must be at least NPTS.

NPTS

Number of given (X,Y) points.

IP2

Work space of dimension at least (NPTS +1).

On Output**IP**

Array in which indices of points in convex hull are returned arranged in counter-clockwise order. Dimension of IP in calling program must be at least (NPTS + 1).

IER

Error parameter.

= 0 Convex hull successfully identified.

Non-fatal errors.

= 1 Duplicate point removed.

Fatal errors.

= 32 Require at least 4 distinct points.

= 33 All points are in a line.

= 34 All $X(I) = XMIN$.

= 35 No interior points.

COMMON BLOCKS

TRIANG1 of length $12_8 = 10_{10}$

ALGORITHM

The routine first uses SORT1 to perform a major sort on increasing $X(I)$ coupled with a minor sort on decreasing $Y(I)$. Duplicate points are rejected. PVEC converts the resulting linked list into the permutation vector, IP. The range of variation of $X(I)$ and $Y(I)$ are determined and scale factors derived. The points with $X = XMIN$ are accepted as being on the counter-clockwise hull since they are already sorted in order of decreasing Y . Suppose that points $IP(1), IP(2), \dots, IP(I1)$ have already been identified as lying on the hull. Then taking $I2 = I1+1, I3 = I1+2$, the points $IP(I1), IP(I2)$ and $IP(I3)$ are tested for counter-clockwise order. If the test is positive, $I3$ is advanced by one and the test repeated. If the three points are in clockwise order, $IP(I2)$ and $IP(I3)$ are interchanged before advancing $I3$.

ALGORITHM
(continued)

This continues until the end of the permutation vector is reached. IP(I2) is then accepted as the next point on the hull. When the first three points on the hull have been identified, the first point is added to the end of the permutation vector so that it may later be identified as closing the hull.

SPACE REQUIRED

$531_8 = 345_{10}$

TIMING

Time to set up the convex hull for a set of 100 data points was 21 msec.

PORTABILITY

This routine is portable.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

SUBROUTINE TMESH (X,Y,NPTS,IP,LINE,ITRI,IER)**DIMENSION OF ARGUMENTS**

X(NPTS),Y(NPTS,IP(NPTS+1),LINE(5,NL),ITRI(3,NT)), where

NL = number of lines = $2*NB+3*(NI-1)$

NT = number of triangles = $NB+2*(NI-1)$

NB = number of boundary points

NI = number of interior points

LATEST REVISION

May 1974

PURPOSE

Given the convex hull of a two dimensional set of points, this routine determines an initial triangulation.

USAGE

CALL TMESH (X,Y,NPTS,IP,LINE,ITRI,IER)

ARGUMENTS**On Input**

X

Array of X coordinates of NPTS given points. Dimension of X in calling program must be at least NPTS.

Y

Array of Y coordinates of NPTS given points. Dimension of Y in calling program must be at least NPTS.

NPTS

Number of given (X,Y) data points.

On Input
(continued)

IP

Array containing the indices of the (NB + 1) points in the convex hull arranged in counterclockwise order IP(NB+1) = IP(1) to close the boundary. Dimension of IP in call program must be at least (NPTS+1).

On Output

LINE

Two dimensional array in which information defining the NL lines of the triangulation are returned.

LINE(1,I) = index of first point of Ith line

LINE(2,I) = index of second point

The Ith line is conventionally directed from LINE(1,I) to LINE(2,I).

LINE(3,I) = index of triangle to left of line

LINE(4,I) = index of triangle to right of line

LINE(5,I) = scaled line length.

The array LINE must be dimensioned (5,NLL) in the calling program, where NLL is at least NL.

ITRI

Two dimensional array in which information defining the NT triangles is returned.

ITRI(1,K) = index of line forming first side of triangle,K.

ITRI(2,K) = index of line forming second side.

ITRI(3,K) = index of line forming third side.

The sides are taken in counterclockwise order with line vectors pointing counterclockwise. If a line vector is pointing in the wrong direction, this fact is indicated by taking the index to be negative. The dimension of ITRI in calling program must be at least (3,NT).

IER

Error parameter.

= 0 Triangulation successfully completed.

Non-fatal errors.

= 2 Point lies on boundary but was not included in boundary.
Point now omitted.

On Output

Fatal errors.

= 36 Interior point not contained in any triangle.

= 37 Data lists bad.

COMMON BLOCKS

TRIANG of length $12_8 = 10_{10}$

ALGORITHM

The centroid of all interior points is calculated and the nearest data point, JC, is identified. Parameters describing each boundary line and the lines joining each boundary point to JC are entered in array LINE (see above). Likewise parameters describing the triangles enclosed by these lines are entered in array ITRI. STEST is then called to check whether any interior lines can be improved. The remaining interior points are then processed successively. The triangle containing the point is identified and three new lines are constructed by joining the point to the vertices of the triangle. The new lines are added to array LINE. Three new triangles are thus created while the original is destroyed. These are added to ITRI, one of them overwriting the parameters for the original triangle. The lines which defined the original triangle are then tested for improvement. This process continues until all points are included in the triangulation.

SPACE REQUIRED

$1645_8 = 933_{10}$

TIMING

Time to set up initial triangulation for 100 data points was 254 msec.

PORTABILITY

One machine dependent constant is defined by a clearly commented data statement.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, SQRT

SUBROUTINE TMESHI (X,Y,LINE,ITRI,IER)

DIMENSION OF
ARGUMENTS

X(NPTS),Y(NPTS),LINE(5,NL),ITRI(3,NT), where

NPTS = total number of data points.

NL = number of lines = $2*NB+3*(NI-1)$ NT = number of triangles = $NB+2*(NI-1)$

NB = number of boundary points

NI = number of interior points

LATEST REVISION

May 1974

PURPOSE

To perform iterative improvement of the initial triangulation set up by TMESH.

USAGE

CALL TMESHI (X,Y,LINE,ITRI,IER)

ARGUMENTS

On Input

X

Array of X coordinates of NPTS given points. Dimension of X in calling program must be at least NPTS.

Y

Array of corresponding Y coordinates. Dimension of Y in calling program must be at least NPTS.

On Input
(continued)**LINE**

Two dimensional array parameterizing the lines of the initial triangulation produced by TMESH. (See TMESH writeup for details.) Dimension of LINE in calling program is (5,NLL) where $NLL \geq NL$.

I TRI

Two dimensional array parameterizing the triangles of the initial triangulation (see TMESH). Dimension of I TRI in calling program must be at least (3,NT).

On Output**LINE, I TRI**

On output the contents of these two arrays have been modified to describe the improved triangulation produced by TMESHI.

IER

Error parameter.

= 0 Iterative improvement successfully completed.

Non-fatal error.

= 3 Iterative improvement terminated on counter.

COMMON BLOCKS

TRIAN1 of length $12_8 = 10_{10}$.

TRIAN2 of length 4.

ALGORITHM

Initially a flag is set on each interior line indicating that they need to be tested. STEST is then applied to each flagged line. If the test is negative, the flag is removed. If it is positive the line is flipped (see STEST) and its flag is removed. At the same time, flags are set on the four neighboring lines. This process is continued until either all lines are unflagged or a maximum of $3*NL$ applications of STEST have been made, whichever occurs first.

The iterative procedure usually converges well before the counter limit is reached.

SPACE REQUIRED $157_8 = 111_{10}$

TIMING Time for iterative improvement of triangulation for 100 data points was 21 msec.

PORTABILITY This routine is portable.

**REQUIRED RESIDENT
ROUTINES** ULIBER

SUBROUTINE SORT1 (LIST,IHEAD,N,LAB,L1,L2,ITEST)

DIMENSION OF ARGUMENTS

LIST(N),IHEAD(N)

LATEST REVISION

May 1974

PURPOSE

To sort a set of records according to criteria provided by the user. The resulting sequence is described by a linked chain.

USAGE

CALL SORT1 (LIST,IHEAD,N,LAB,L1,L2,ITEST)
CALL SORT2 (LIST,IHEAD,N,LAB,L1,L2,ITEST)

The user's sort criteria are applied in the calling program as follows:

LOGICAL LAB

```
C   INITIALIZATION
C   CALL SORT1(LIST,IHEAD,N,LAB,L1,L2,ITEST)
C   BEGIN SORT PROPER
10  CONTINUE
    CALL SORT2(LIST,IHEAD,N,LAB,L1,L2,ITEST)
    IF (LAB) GO TO 20
C   USER CODE TO SET ITEST APPEARS HERE
C   ITEST.LT.0 IF RECORD L1 SHOULD PRECEDE L2
C   ITEST = 0 IF DON'T CARE
C   ITEST.GT.0 IF L1 SHOULD FOLLOW L2.
    GO TO 10
20  CONTINUE
```

To process items in order, do the following:

```

      I = L1
30   IF (I.EQ.0) GO TO 40
C    USER CODE TO PROCESS ITEM
      I = LIST(I)
      GO TO 30
40   CONTINUE

```

ARGUMENTS

On Input

N

Number of items to be sorted.

IHEAD

Work area of dimension at least N.

ITEST

Parameter set by user sort code.

< 0 if record L1 to precede L2

= 0 if don't care

> 0 if L1 to follow L2.

(See L1, L2 below).

On Output

LIST

Array in which the sorted sequence is returned as a linked chain. The index of the first item is returned in L1, the index of the next item will appear in LIST(L1) and so on until the chain is terminated by a zero element in LIST. Dimension of LIST in calling program must be at least N.

LAB

Logical variable with values

= .TRUE. if sort is completed

= .FALSE. if sort incomplete

On Output
(continued)

L1, L2

Indices of two items to be sorted by user code.

ENTRY POINTS

SORT1, SORT2

COMMON BLOCKS

None

ALGORITHM

All records are scanned to find existing ascending and descending sequences. The heads of these are stored in the array IHEAD, while the sequences appear as linked chains in the array LIST. These sequences are then merged to produce a single linked chain.

SPACE REQUIRED

$314_8 = 204_{10}$

PORTABILITY

This routine is portable.

**REQUIRED RESIDENT
ROUTINES**

None

SUBROUTINE PVEC (LIST,HEAD)**DIMENSION OF
ARGUMENTS**

LIST(N) where N = number of items.

LATEST REVISION

May 1974

PURPOSE

To convert a chain of items in a list to a permutation vector.

USAGE

CALL PVEC (LIST,HEAD)

ARGUMENTS**On Input**

LIST

Array containing linked chain of items. Dimension of LIST in calling program must be at least N where N is number of items.

HEAD (integer)

Head of list in array LIST.

On Output

LIST

The linked chain input in this array is replaced by the equivalent permutation vector.

COMMON BLOCKS

None

ALGORITHM

The linked chain in LIST is first replaced by an inverse permutation vector, which in turn is transformed to the required permutation vector.

SPACE REQUIRED

$74_8 = 60_{10}$

PORTABILITY

This routine is portable.

**REQUIRED RESIDENT
ROUTINES**

None

SUBROUTINE STEST (X,Y,LINE,ITRI,LL,FLIP)**DIMENSION OF ARGUMENTS**

X(NPTS),Y(NPTS),LINE(5,NL),ITRI(3,NT) where

NPTS = total number of data points.

NL = number of lines = $2*NB+3*(NI-1)$.

NT = number of triangles = $NB+2*(NI-1)$.

NB = number of boundary points.

NI = number of interior points.

LATEST REVISION

May 1974

PURPOSE

To test whether the triangulation can be improved by exchanging the given interior line, LL, for the other diagonal of the quadrilateral of which LL is a diagonal. If so, the exchange is performed.

USAGE

CALL STEST (X,Y,LINE,ITRI,LL,FLIP)

ARGUMENTS**On Input**

X

Array of X coordinates of given data points. Dimension of X in calling program must be at least NPTS where NPTS is the total number of points.

Y

Array of corresponding Y coordinates. Dimension of Y in calling program must be at least NPTS.

On Input
(continued)**LINE**

Two dimensional array parameterizing the lines of the current triangulation (see TMESH writeup for details). Dimension of LINE in calling program must be at least (5,NL).

I TRI

Two dimensional array parameterizing the triangles of the current triangulation (see TMESH). Dimension of I TRI in calling program must be at least (3,NT).

LL

Index of interior line to be tested.

On Output**FLIP (logical)**

= .TRUE. Line was changed.

= .FALSE. Line unchanged.

COMMON BLOCKS

TRIAN1 of length 12₈ = 10₁₀.

TRIAN2 of length 4.

ALGORITHM

The two triangles on either side of the given interior line are identified. If these together form a convex quadrilateral, exchange is possible. A quantity proportional to the sine of the minimum interior angle of the two triangles is determined. The same quantity is then evaluated for the two triangles produced by exchanging the given line for the other diagonal of the quadrilateral. If these triangles have a larger minimum angle, the exchange is performed by making appropriate modifications to the arrays describing the triangulation. Otherwise the line is left unchanged.

SPACE REQUIRED $472_8 = 314_{10}$

PORTABILITY Two machine dependent constants are defined by clearly commented data statements.

REQUIRED RESIDENT
ROUTINES SQRT

FUNCTION LTEST (SX,SY,DX,DY)

LATEST REVISION May 1974

PURPOSE

This function routine evaluates the cross product of the two vectors (SX,SY) and (DX,DY) and uses it to distinguish between three situations

- Angle between (SX,SY) and (DX,DY) $< \pi$
- (DX,DY) is a multiple of (SX,SY) between 0. and 1.
- Angle between (SX,SY) and (DX,DY) $\geq \pi$ or (DX,DY) is a multiple of (SX,SY) $\leq 0.$ or $\geq 1.$

The vectors are assumed scaled to order unity. If the cross product is less than machine accuracy, it is treated as zero.

USAGE

LT = LTEST (SX,SY,DX,DY)

LT = 1 if angle between (SX,SY) and (DX,DY) is $< \pi$
 = 0 if (DX,DY) is multiple of (SX,SY) between 0. and 1.
 = -1 otherwise

ARGUMENTS

On Input

SX

X-component of first vector of cross product.

SY

Y-component of first vector of cross product.

DX

X-component of second vector of cross product.

DY

Y-component of second vector of cross product.

COMMON BLOCKS

None

SPACE REQUIRED

$74_8 = 60_{10}$

PORTABILITY

One machine dependent constant is defined by a clearly commented data statement.

**REQUIRED RESIDENT
ROUTINES**

None

LINEAR ALGEBRA

BDSLV

BND3

CHLSLV

EIGCFA

EIGHFS

EIGRFA

EIGSFM

EIGSFS

EIGSTM

EIGSTS

HSHSLV

INVMTX

LINEQSV

SUPRLS

SVDSLV

TRDI

TRDIP

SUBROUTINE BDSLV (N,M,S,KS,B,X,WORK,NFLAG)**DIMENSION OF ARGUMENTS**

S(N,2M+1),B(N),X(N),WORK(N*(2M+3)+M*(N-1))

LATEST REVISION

November 1973

PURPOSE

BDSL

ACCESS CARDS*FORTRAN,S=ULIB,N=BDSL
*COSY**USAGE**

CALL BDSLV (N,M,S,KS,B,X,WORK,NFLAG)

ARGUMENTS**On Input**

N

The number of unknowns.

M

The integer such that $A(i,j) = 0$ when $|i-j| > M$. The band width of A is then $2M+1$. (Example: for a tridiagonal matrix, $M = 1$.)

On Input
(continued)

S

A two dimensional array containing the non-zero elements of the matrix A in a special, packed format. The way the row I of the matrix A is stored in row I of S is described by:

1. When $I = 1, 2, \dots, M$; $S(I, J) = A(I, J)$
for $J = 1, 2, \dots, 2M+1$
2. When $I = M+1, M+2, \dots, N-M$; $S(I, J) = A(I, I-M-1+J)$
for $J = 1, 2, \dots, 2M+1$
3. When $I = N-M+1, N-M+2, \dots, N$; $S(I, J) = A(I, I-M-1+J)$
for $J = 1, 2, \dots, N+M+1-I$

(If the user does not want to save S, S can be set to WORK.)

KS

- = 0 if this is the initial call to BDSLV with a given coefficient matrix A.
- = 1 if the coefficient matrix A is unchanged from the previous call.

B

The right hand side of the system of equations.

WORK

An array which must be provided for work space. WORK must be dimensioned at least $N*(2M+3)+M*(N-1)$.

On Output

X

An array which contains the solution to the system of equations. (If the user does not want to save B, X can be set to B.)

WORK

Contains intermediate values which must not be destroyed if BDSLV is to be called again with $KS = 1$.

NFLAG

An error flag which indicates an unsuccessful attempt to solve the system of equations.

The flag is set to:

= 1 when a zero pivot element is encountered. The following message is also printed out

ZERO PIVOT ELEMENT ENCOUNTERED, A IS SINGULAR.

= 2 when a zero row is found in the input matrix. The following message is also printed out.

ZERO ROW IN INPUT MATRIX.

= 3 when the band width ($2M+1$) is greater than N . The following message is also printed out.

BAND WIDTH IS LARGER THAN N .

Control is returned to the calling program in each case.

ENTRY POINTS

BDSL.V, BNDX

SPECIAL CONDITIONS

If the system is singular, a solution may not exist. A message is printed out in BNDX, NFLAG is set, and control is returned to the calling program.

COMMON BLOCKS

None

I/O

See "Special Conditions"

PRECISION

Single

REQUIRED ULIB ROUTINES

None

SPECIALIST Nancy Werner, NCAR, Boulder, Colorado 80303

HISTORY Standardized November 1973

ALGORITHM The system of equations is solved using Gaussion elimination with row-scaling and partial pivoting. Reference: *Computer Solution of Linear Algebraic Systems*, Forsythe and Moler.

SPACE REQUIRED $715_8 = 461_{10}$ locations

ACCURACY Usually equal to machine precision, unless the matrix is ill-conditioned. For a discussion of the condition of a matrix, see Wilkinson, *The Algebraic Eigenvalue Problem*.

TIMING The running time is very roughly proportional MN.

The approximate running time for a matrix of order 50 with a band width of 3 on the NCAR CDC 7600 is 2 milliseconds when $KS = 0$. If $KS = 1$, the running time is .7 milliseconds.

PORTABILITY There are no machine dependent constants.

REQUIRED RESIDENT ROUTINES None

Method

Subroutine BDSLV is used to solve a system of equations in which the coefficient matrix is a band matrix.

Example: Given the system of equations

$$\begin{array}{rcl}
 5X_1 - 4X_2 + X_3 & = & Y_1 \\
 -4X_1 + 6X_2 - 4X_3 + X_4 & = & Y_2 \\
 X_1 - 4X_2 + 6X_3 - 4X_4 + X_5 & = & Y_3 \\
 X_2 - 4X_3 + 6X_4 - 4X_5 + 1X_6 & = & Y_4 \\
 X_3 - 4X_4 + 6X_5 - 4X_6 & = & Y_5 \\
 X_4 - 4X_5 + 5X_6 & = & Y_6
 \end{array}$$

the input to BDSLV would be as follows:

$$\begin{array}{rcl}
 N & = & 6 \\
 M & = & 2 \\
 B(I) = Y_I & & I = 1, 2, \dots, 6
 \end{array}$$

The original coefficient matrix is

$$A = \begin{bmatrix} 5 & -4 & 1 & 0 & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & 0 & 1 & -4 & 5 \end{bmatrix}$$

This matrix is stored in S in the packed form

$$S = \begin{bmatrix} 5 & -4 & 1 & X & X \\ -4 & 6 & -4 & 1 & X \\ 1 & -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 & X \\ 1 & -4 & 5 & X & X \end{bmatrix}$$

where X represents an element which is not specified.
(These elements are set to zero in BDSLV.)

3.BDSL.V.6

Method

(continued)

The dimension of WORK must be (52).

For the first call to BDSLV, KS must = 0. If BDSLV is called again and the S array has not been changed, then set KS = 1.

SUBROUTINE BND3 (N,A,B,C,Y,X,WORK,NFLAG)

DIMENSION OF ARGUMENTS A(N),B(N),C(N),Y(N),X(N),WORK(3*N-6)

LATEST REVISION September 1973

PURPOSE BND3 computes a solution to a system of equations whose matrix is tridiagonal.

ACCESS CARDS *FORTRAN,S=ULIB,N=BND3
 *COSY

USAGE CALL BND3 (N,A,B,C,Y,X,WORK,NFLAG)

ARGUMENTS

On Input N
 The number of equations.

 A
 The subdiagonal of the matrix is stored in locations A(2) through A(N).

 B
 The diagonal of the matrix is stored in locations B(1) through B(N).

 C
 The super-diagonal of the matrix is stored in locations C(1) through C(N-1).

On Input
(continued)

Y

The right-hand side of the equations is stored in Y(1) through Y(N).

WORK

An array which must be provided for work space. WORK must be dimensioned at least $3*(N-2)$.

On Output

X

An array which contains the solution to the system of equations. If the user does not want to save the array Y, then X may be set to Y.

NFLAG

= 0 no error
= 1 singular matrix.

ENTRY POINTS

BND3

SPECIAL CONDITIONS

If the system is singular, NFLAG is set to 1; a message is printed out by ULIBER; and execution proceeds. At present this subroutine tests only for a zero denominator. The system may be essentially singular and still not satisfy this test.

COMMON BLOCKS None

I/O See "Special Conditions."

PRECISION Single

**REQUIRED ULIB
ROUTINES** None

SPECIALIST Nancy Werner, NCAR, Boulder, Colorado 80302

LANGUAGE FORTRAN

HISTORY Standardized September, 1973

ALGORITHM A system of equations described by a tridiagonal matrix is
solved using Gaussian elimination with partial pivoting.

SPACE REQUIRED (242)₈ = (162)₁₀ locations

At this stage, the element a_{i+1} must be zeroed using either Method 1 or Method 2, depending on the magnitude of this element.

Method 1: If $|BI| \geq |a_{i+1}|$ then set

$$D_i = CI/BI; \quad F_i = 0; \quad W_i = YI/BI$$

Then equations $i-1$ through $i+1$ have the appearance:

$$1 \quad D_{i-1} \quad F_{i-1} \quad = \quad W_{i-1}$$

$$0 \quad 1 \quad D_i \quad F_i \quad = \quad W_i$$

$$a_{i+1} \quad b_{i+1} \quad c_{i+1} \quad = \quad Y_{i+1}$$

Multiply the i^{th} equation by a_{i+1} and subtract it from the $(i+1)^{\text{st}}$.

$$1 \quad D_{i-1} \quad F_{i-1} \quad = \quad W_{i-1}$$

$$1 \quad D_i \quad F_i \quad = \quad W_i$$

$$0 \quad b_{i+1}^{-a_{i+1}} D_i \quad c_{i+1}^{-a_{i+1}} F_i \quad = \quad Y_{i+1}^{-a_{i+1}} W_i$$

Now denote $BI = b_{i+1}^{-a_{i+1}} D_i$

$$CI = c_{i+1}^{-a_{i+1}} F_i$$

$$YI = Y_{i+1}^{-a_{i+1}} W_i$$

and thus the i^{th} step in the sequence is completed.

Purpose
(continued)

Method 2: If $|BI| < |a_{i+1}|$ then interchange the rows before elimination.

$$1 \quad D_{i-1} \quad F_{i-1} \quad = \quad W_{i-1}$$

$$a_{i+1} \quad b_{i+1} \quad C_{i+1} \quad = \quad Y_{i+1}$$

$$BI \quad CI \quad = \quad YI$$

Then divide the middle row by a_{i+1} and set

$$D_i = \frac{b_{i+1}}{a_{i+1}}; \quad F_i = \frac{c_{i+1}}{a_{i+1}}; \quad W_i = \frac{Y_{i+1}}{a_{i+1}}$$

$$1 \quad D_{i-1} \quad F_{i-1} \quad = \quad W_{i-1}$$

$$0 \quad 1 \quad D_i \quad F_i \quad = \quad W_i$$

$$BI \quad CI \quad = \quad YI$$

Then multiply the middle row by BI and subtract from the bottom row:

$$1 \quad D_{i-1} \quad F_{i-1} \quad = \quad W_{i-1}$$

$$0 \quad 1 \quad D_i \quad F_i \quad = \quad W_i$$

$$0 \quad CI - BI \cdot D_i \quad -BI \cdot F_i \quad = \quad YI - BI \cdot W_i$$

Denote the non-zero elements of the last row by BI, CI and YI which completes the i^{th} stage of the elimination.

After $n-2$ such operations, the lower right hand of the matrix has the form:

$$\begin{array}{r} \cdot \\ \cdot \\ \cdot \\ 1 \quad D_{n-2} \quad F_{n-2} = W_{n-2} \\ 0 \quad B_I \quad C_I = Y_I \\ \\ 0 \quad a_n \quad b_n = Y_n \end{array}$$

The 2×2 matrix is solved using Cramers rule for the last two unknowns x_{n-1} , x_n .

The remaining unknowns are determined by a backward sweep

$$x_k = W_k - D_k \cdot x_{k+1} - F_k \cdot x_{k+2} \quad (k = n-2, n-3, \dots, 1)$$

The symbols used in the Fortran program are defined as follows:

$$D_i = \text{WORK}(I)$$

$$F_i = \text{WORK}(I1)$$

$$W_i = \text{WORK}(I2)$$

SYMMETRIC POSITIVE DEFINITE LINEAR SYSTEM SOLVER**LATEST REVISION**

December 1973

PURPOSE

CHLSLV calculates the solution vector for a real, symmetric, positive definite linear system $A*X = b$. CHLSLV contains four subroutines; CHLSKY, CHSLV1, CHSLV2 and CHIMPR.

CHLSKY decomposes a real symmetric positive definite matrix A into the product of a lower triangular matrix and its transpose, such that $A = L*L^T$. (Cholesky decomposition)

CHSLV1 calculates the solution vector Y of the lower triangular linear system $L*Y = b$.

CHSLV2 calculates the solution vector of the upper triangular linear system $L^T*X = Y$.

CHIMPR performs iterative improvement on the calculated solution vector X .

ACCESS CARDS

*FORTRAN,S=ULIB,N=CHLSLV
*COSY

SPACE REQUIRED

571₈ = 377₁₀

USAGE

The four subroutines of the file should be called in the following sequence:

CALL CHLSKY (A,L,D,MA,ML,M,ISW)

CALL CHSLV1 (L,D,B,Y,ML,M)

CALL CHSLV2 (L,D,Y,X,ML,M)

CALL CHIMPR (A,L,D,B,X,ML,MA,M,ITS,TEMP)

ARGUMENTS

On Input
for CHLSKY

A

Contains the elements of the symmetric positive definite matrix. A has dimension (MA,M). A need only contain the upper triangular elements of the positive definite matrix.

MA

Contains the row dimension of the matrix A as described in the dimension statement of the calling program.

M

Contains the actual row dimension of A used in CHLSKY.

ML

Contains the row dimension of the matrix L as declared in the dimension statement of the calling program.

On Output
for CHLSKY

L

Is a matrix with dimension (ML,M) which contains the strictly lower triangular elements of the Cholesky decomposition of A. L is a real (not integer) variable.

If A is entered twice in the parameter list, replacing L, then on output, A will contain the elements of the matrix A as described in input and of L described above.

D

Is a vector with dimension M which contains the inverse of the diagonal elements of the decomposition of A.

ISW

Is an integer error flag.

= 1 if A is not positive definite.

= 0 otherwise.

On Input
for CHSLV1

L

Is a matrix with dimension (ML,M). L contains the elements of the matrix L described in output for CHLSKY.

D

Is a vector with dimension M. D contains the elements of the vector D described in output for CHLSKY.

B

Is a vector with dimension M. On input, B contains the elements of the right hand side of the symmetric positive definite system.

ML

Is an integer which contains the row dimension of the matrix L as declared in the dimension statement of the calling program.

M

Is an integer which contains the row dimension of L actually used in CHSLV1.

On Output
for CHSLV1

Y

Is a vector with dimension M. On output, Y contains the solution vector of the lower triangular system $L*Y = b$.

3.CHLSLV.4

On Input
for CHSLV2

L

Is a matrix with dimension (ML,M). On input, L contains the elements of the matrix L described in output for CHLSKY.

D

Is a vector with dimension M. On input, D contains the elements of the vector D described in output for CHLSKY.

Y

Is a vector with dimension M. On input, Y contains the elements of the vector Y described in output for CHSLV1.

ML

Is an integer input variable which contains the row dimension of the matrix L as declared in the dimension statement of the calling program.

M

Is an integer input variable which contains the actual row dimension of L used in CHSLV2.

On Output
for CHSLV2

X

Is a vector variable with dimension M which contains the solution for the upper triangular system $L^T * X = Y$.

On Input
for CHIMPR

A

Is a matrix with dimension (MA,M). On input, A contains the elements of the matrix A described in input for CHLSKY.

L

Is a matrix with dimension (ML,M). On input, L contains the elements of the matrix L described in output for CHLSKY.

D

Is a vector with dimension M. On input, D contains the elements of the vector D described in output for CHLSKY.

B

Is a vector with dimension M. On input, B contains the elements of the vector B described in input for CHSLV1.

X

Is a vector with dimension M. On input, X contains the elements of the vector X described in output for CHSLV2.

ML

Is an integer which contains the row dimension of the matrix L as declared in the dimension statement of the calling program.

MA

Is an integer which contains the row dimension of the matrix A as declared in the dimension statement of the calling program.

M

Is an integer which contains the row dimension of A actually used in CHIMPR.

ITS

Is an integer which contains the maximum number of iterations for iterative improvement (e.g., ITS = 10).

TEMP

Is a real vector array which is used internally for working storage. It must have dimension at least 2*M.

On Output
for CHIMPR

X

Contains the elements of the improved solution vector.

ENTRY POINTS

CHLSKY, CHSLV1, CHSLV2, CHIMPR

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

- In CHLSKY, if the matrix A is not positive definite, the subroutine prints the message

DIAGONAL ELEMENT NON-POSITIVE IN SUBROUTINE CHLSKY.

- In CHIMPR, if the algorithm does not converge before it reaches the value of the variable ITS, the subroutine prints the message

EXCEED ITERATIONS IN CHIMPR.

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Written in 1972, standardized December 1973.

ALGORITHM

CHLSLV calculates the solution vector X for the linear system

$$A*X = b$$

where A is real, symmetric and positive definite.

1. Subroutine CHLSKY decomposes the matrix A into the product of a lower triangular matrix and its transpose such that

$$A = L*L^T .$$

Thus, the system

$$A*X = b$$

becomes

$$L*L^T*X = b .$$

2. Subroutine CHSLV1 calculates the solution vector Y for the lower triangular linear system

$$L*Y = b .$$

3. Subroutine CHSLV2 calculates the solution vector X for the upper triangular linear system

$$L^T*X = Y .$$

This X is the calculated solution to the original system

$$A*X = b.$$

4. Subroutine CHIMPR performs iterative improvement on the calculated solution vector X.

ACCURACY

The accuracy of CHLSLV depends on the conditioning of the matrix A and the error in the measurement of the vector B.

TIMING

The timing for CHLSLV is proportional to the quantity

$$\frac{M^3}{6} + M^2 .$$

On the CDC 7600, with M = 50, CHLSLV takes 37 milliseconds.

PORTABILITY

CHLSLV is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

REQUIRED RESIDENT
ROUTINES

SQRTF, ULIBER

EIGCFA**SUBROUTINE EIGCFA (NDIM,N,M,AR,AI,WR,WI,ZR,ZI,SELECT,IERR,JERR,WORK)****DIMENSION OF
ARGUMENTS**

AR(NDIM,N),AI(NDIM,N),WR(N),WI(N),ZR(NDIM,M),ZI(NDIM,M),
SELECT(N),WORK(N*(4*NDIM+6))

LATEST REVISION

March 1974

PURPOSE

EIGCFA computes all the eigenvalues and selected
eigenvectors of a general complex matrix A.

ACCESS CARDS

*FORTRAN,S=ULIB,N=EIGCFA
*COSY

USAGE

CALL EIGCFA (NDIM,N,M,AR,AI,WR,WI,ZR,ZI,SELECT,IERR,JERR,
WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension in the calling program of the two-dimensional arrays AR, AI, ZR, ZI.

N

The order of the complex matrix A.

M

The maximum number of eigenvectors to be found.

AR, AI

Arrays containing the real and imaginary parts, respectively, of the matrix A. The row (first) dimension of these arrays must be equal to NDIM.

SELECT

A logical array of dimension N indicating which eigenvectors are desired. Set SELECT(K)=.TRUE. if the eigenvector corresponding to the eigenvalue $WR(K)+SQRT(-1)*WI(K)$ is desired; otherwise, set it to .FALSE. . If more than M elements of SELECT are .TRUE. , JERR is set to N+1, control is returned to the user after the eigenvalue computation, and no eigenvectors are computed. If all the eigenvalues cannot be found, determination of eigenvectors will be attempted only when every entry of SELECT is .TRUE.

WORK

An array which must be provided for work space. WORK must be dimensioned at least $N*(4*NDIM+6)$ if any eigenvectors are requested; if M is set to zero (no eigenvectors requested) it need only be dimensioned $N*(4*NDIM+3)$.

On Output

WR, WI

Arrays of dimension N containing the real and imaginary parts, respectively, of the eigenvalues. The eigenvalues will be ordered by increasing magnitude with those of equal magnitude being ordered by increasing argument between 0 and 2π . If all the eigenvalues cannot be found, IERR is set to J indicating that N-J have been found and are stored in WR(I), WI(I) for $I=J+1, J+2, \dots, N$.

ZR, ZI

Two-dimensional arrays whose columns contain the real and imaginary parts, respectively, of the computed eigenvectors. The row dimension of the arrays in the calling program must be equal to NDIM and the column dimension at least M. The eigenvectors are ordered according to the SELECT array and are returned with the component of largest magnitude normalized to 1. If a desired eigenvector cannot be found, the zero vector is returned. If no eigenvectors are desired, these arguments may be dummy variables.

SELECT

If J eigenvalues cannot be found and all the eigenvectors were desired, then SELECT(I) for $I=1, 2, \dots, J$ are set to .FALSE.

IERR

An error flag related to computing the eigenvalues.

= 0, if no error.

= J, if only N-J eigenvalues were found. These eigenvalues are stored in WR(I), WI(I) for $I=J+1, J+2, \dots, N$. If M is less than N, no attempt is made to find any eigenvectors.

JERR

An error flag related to computing the eigenvectors.

= 0, if no error.

= J, if the iteration for the Jth eigenvector failed to converge. The zero vector is returned. Only the latest failure is recorded by JERR.

=N+1, if M is less than the number of .TRUE entries of SELECT. An attempt will be made to find the eigenvalues, but not the eigenvectors.

ENTRY POINTS EIGCFA, CBAL, COMHES, COMLR, CINVIT, COMBAK, CBABK2

SPECIAL CONDITIONS All the eigenvalues are found unless an error message indicates otherwise. If this occurs, no eigenvectors will be found unless all the eigenvectors were requested, in which case the eigenvectors will be found corresponding to those eigenvalues which were obtained. The SELECT array is set to .FALSE. for the missing eigenvalues. Those eigenvectors indicated by the SELECT array are found unless an error message indicates otherwise. If more than one error occurs, JERR will be set to indicate the last occurring error.

COMMON BLOCKS None

I/O See "Special Conditions"

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Nancy Werner, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY

This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM

First CBAL is called to balance the complex matrix and isolate eigenvalues whenever possible. Then COMHES is called to reduce the submatrix specified by CBAL to upper Hessenberg form by stabilized elementary similarity transformations. COMLR is then called to find the eigenvalues of a complex upper Hessenberg matrix by the modified LR method. CINVIT is called to find those eigenvectors of a complex upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration. COMBAK and CBABK2 then form the desired eigenvectors by back transforming those of the corresponding upper Hessenberg matrix determined by COMHES. These eigenvectors are then normalized so that the component of largest magnitude is 1.

Finally, the eigenvalues are arranged in ascending order of magnitude; those with equal magnitude are ordered by increasing real part. The selected eigenvectors are arranged to correspond to the ordered eigenvalues.

SPACE REQUIRED

4715₈ = 2509₁₀

ACCURACY

The eigenvalues and eigenvectors will usually be determined to within an absolute error of $10^{**(-14)}$.

TIMING

The approximate execution time for a matrix of order 20 on the CDC 7600 is as follows:

All values, all vectors = 160 milliseconds

All values, no vectors = 96 milliseconds

The execution time necessary to obtain all the eigenvalues is roughly proportional to N^2 . The execution time necessary to obtain each eigenvector is proportional to N^2 .

PORTABILITY

There is a machine dependent constant defined and used in the subroutines COMLR and CINVIT. It is also defined and used in EIGCFA to order the eigenvalues

"MACHEP = 2.**(-47)"

**REQUIRED RESIDENT
ROUTINES**

ATAN2

SUBROUTINE EIGHFS (NDIM,N,AR,AI,WR,ZR,ZI,EPST,XLB,UB,NMX,KF,IERR,WORK)

DIMENSION OF ARGUMENTS AR(NDIM,N),AI(NDIM,N),WR(NMX),ZR(NDIM,NMX),ZI(NDIM,NMX),
WORK(N*(2*NDIM+10)+NMX)

LATEST REVISION January 1973

PURPOSE EIGHFS determines all the eigenvalues of a complex,
Hermitian matrix within a specified interval.
Optionally the eigenvectors corresponding to the
computed eigenvalues are found.

ACCESS CARDS *FORTRAN,S=ULIB,N=EIGHFS
*COSY

USAGE CALL EIGHFS (NDIM,N,AR,AI,WR,ZR,ZI,EPST,XLB,UB,NMX,KF,
IERR,WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension of the two dimensional arrays AR, AI, ZR and ZI in the calling program.

N

The order of the complex, Hermitian matrix A.

AR, AI

Arrays containing the real and imaginary parts, respectively, of the matrix A. The row (first) dimension of these arrays in the calling program must be equal to NDIM.

EPS1

An absolute error tolerance used in computing the eigenvalues. If EPS1 is non-positive, the routine computes a default value for EPS1 which is usually sufficient for computing the eigenvalues as accurately as possible. If eigenvectors are to be computed, the eigenvalues must have been calculated with high relative precision. Setting EPS1 non-positive will insure that they are computed accurately enough. See the comments under "Accuracy".

XLB, UB

Endpoints of the interval to be searched for eigenvalues. If $XLB \geq UB$, IERR is set to 1 and control is returned to the calling program.

NMX

An integer specifying an upper bound on the number of eigenvalues in the interval (XLB, UB). If more than NMX eigenvalues are found, execution ceases and control is returned to the calling program with M set equal to the actual number of eigenvalues found and with no eigenvalues specified in the output vector.

KF

A flag which is set to indicate if the corresponding eigenvectors are desired.

- = 0 no eigenvectors are desired.
- = 1 all eigenvectors for those eigenvalues found in the specified interval are desired.

WORK

An array which must be provided for workspace. WORK must be dimensioned at least $N*(2*NDIM+10)+NMX$ if eigenvectors are requested; if no eigenvectors are requested, it need be dimensioned only $N*(2*NDIM+7)+NMX$.

On Output

WR

An array of dimension NMX which contains the eigenvalues in ascending order.

ZR,ZI

Arrays containing the real and imaginary parts, respectively, of the computed eigenvectors ordered by columns to correspond with the WR array. The eigenvectors are normalized so that the component of largest magnitude is one. The row (first) dimension of these arrays in the calling program must be NDIM and the column dimension must be at least NMX. If the iteration for an eigenvector fails, the zero vector is returned to the user. If eigenvectors are not desired (KF=0), ZR and ZI may be dummy arguments.

EPS1

If EPS1 was originally non-positive, it will be set to its last default value.

On Output
(continued)

IERR

An error flag.

= 0, no error has occurred.

= 1, $XLB \geq UB$

= $3*N+1$, more than NMX eigenvalues have been found in the specified interval. An error message will be printed out indicating this. The actual number found is returned in NMX.

= -R, if the iteration for the eigenvector corresponding to the R^{th} eigenvalue fails to converge in 5 iterations. If this failure occurs for more than one eigenvector, the last occurrence is recorded in IERR. An error message is printed out indicating this.

ENTRY POINTS

EIGHFS, HTRIDI, BISECT, TINVIT, HTRIBK

SPECIAL CONDITIONS

The eigenvalues in a specified interval (XLB, UB) are sought. If the number of eigenvalues in that interval exceeds a maximum number, NMX, an error message will be printed out and no eigenvalues will be found. Control is returned to the calling program.

If $KF \neq 0$, eigenvectors will be sought for all the above eigenvalues. If an iteration for the vector fails to converge, it will be set to zero. An error message will be printed out and computation continued.

COMMON BLOCKS

None

I/O

See "Special Conditions"

PRECISION Single

**REQUIRED ULIB
ROUTINES** None

SPECIALIST Nancy Werner, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM The complex Hermitian matrix is first reduced to a real symmetric tridiagonal matrix by the subroutine HTRIDI using unitary similarity transformations. The eigenvalues of this tridiagonal matrix which lie in a specified interval are found by the subroutine BISECT, using bisection.

If requested, the eigenvectors of the tridiagonal matrix corresponding to these eigenvalues are found by the subroutine TINVIT using inverse iteration. These eigenvectors are then transformed by the subroutine HTRIBK to form the eigenvectors of the original complex Hermitian matrix. Then the eigenvectors are normalized so that the component of largest magnitude is one.

SPACE REQUIRED

 $3135_8 = 1629_{10}$

ACCURACY

Eigenvalues will be computed to within an absolute error given by EPS1 when this parameter is specified to be positive. When EPS1 is specified non-positive, the routine computes a value of EPS1 adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of $2. \times 10^{-47}$ in the matrix elements. A more detailed discussion of the use of EPS1 may be found in the EISPACK write-up.

TIMING

This routine requires $0.0016N^3$ milliseconds to reduce the given matrix to tridiagonal form. Using this form it requires $0.26N^2$ milliseconds to compute all the eigenvalues and associated eigenvectors. These times are for the NCAR CDC 7600.

PORTABILITY

There is a machine dependent constant defined and used in the subroutines BISECT and TINVIT:

```
"MACHEP = 2. \times 10^{-47}"
```

ULIBER, a system resident routine, is used to print error messages. A FORTRAN version of this routine is on ULIB.

REQUIRED RESIDENT
ROUTINES

ULIBER

SUBROUTINE EIGRFA (NDIM,N,MM,AR,WR,WI,ZR,ZI,SELECT,IERR,JERR,WORK)

**DIMENSION OF
ARGUMENTS**

AR(NDIM,N),WR(N),WI(N),ZR(NDIM,MM),ZI(NDIM,MM),SELECT(N),
WORK(N*(N+9)+NDIM*(2*MM+N))

LATEST REVISION

May 1974

PURPOSE

EIGRFA computes all the eigenvalues and selected eigenvectors
of a general real matrix AR.

ACCESS CARDS

*FORTRAN,S=ULIB,N=EIGRFA
*COSY

USAGE

CALL EIGRFA (NDIM,N,MM,AR,WR,WI,ZR,ZI,SELECT,IERR,JERR,WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension in the calling program of the
two-dimensional arrays AR, ZR, ZI.

On Input
(continued)

N

The order of the real matrix AR. N must be less than or equal to NDIM.

MM

The maximum number of eigenvectors to be found.

AR

A real two-dimensional array with row (first) dimension NDIM and column dimension N. AR contains the matrix of order N.

SELECT

A logical array of dimension N indicating which eigenvectors are desired. Set SELECT(K)=.TRUE. if the eigenvector corresponding to the eigenvalue $WR(K)+SQRT(-1)*WI(K)$ is desired; otherwise, set it to .FALSE. . If more than M elements of SELECT are .TRUE. , JERR is set to N+1, control is returned to the user after the eigenvalue computation, and no eigenvectors are computed. If all the eigenvalues cannot be found, determination of eigenvectors will be attempted only when every entry of SELECT is .TRUE. .

WORK

An array which must be provided for work space. WORK must be dimensioned at least $N*(N+9)+NDIM*(2*MM+N)$.

On Output

WR,WI

Arrays of dimension N containing the real and imaginary parts, respectively, of the eigenvalues. The eigenvalues will be ordered by increasing magnitude with those of equal magnitude being ordered by increasing argument between 0 and 2π . If all the eigenvalues cannot be found, IERR is set to J indicating that N-J have been found and are stored in WR(I), WI(I) for $I=J+1, J+2, \dots, N$.

ZR,ZI

Two-dimensional arrays whose columns contain the real and imaginary parts, respectively, of the computed eigenvectors. The row dimension of the arrays in the calling program must be equal to NDIM and the column dimension at least MM. The eigenvectors are ordered according to the SELECT array and are returned with the component of largest magnitude normalized to 1. If a desired eigenvector cannot be found, the zero vector is returned. If no eigenvectors are desired, these arguments may be dummy variables.

SELECT

If J eigenvalues cannot be found and all the eigenvectors were desired, then SELECT(I) for I=1,2,...,J are set to .FALSE. .

IERR

An error flag related to computing the eigenvalues.

= 0 if no error

= J if only N-J eigenvalues were found. These eigenvalues are stored in WR(I), WI(I) for I=J+1,J+2,...,N. If M is less than N, no attempt is made to find any eigenvectors.

JERR

An error flag related to computing the eigenvectors.

= 0 if no error

= J if the iteration for the Jth eigenvector failed to converge. The zero vector is returned. Only the latest failure is recorded by JERR.

= N+1 if M is less than the number of .TRUE. entries of SELECT. An attempt will be made to find the eigenvalues, but not the eigenvectors.

ENTRY POINTS

EIGRFA, BALANC, ORTHES, HQR, INVIT, ORTBAK, BALBAK

SPECIAL CONDITIONS

All the eigenvalues are found unless an error message indicates otherwise. If this occurs, no eigenvectors will be found unless all the eigenvectors were requested, in which case the eigenvectors will be found corresponding to those eigenvalues which were obtained. The SELECT array is set to .FALSE. for the missing eigenvalues. Those eigenvectors indicated by the SELECT array are found unless an error message indicates otherwise. If more than one error occurs, JERR will be set to indicate the last occurring error.

COMMON BLOCKS

None

I/O

See "SPECIAL CONDITIONS".

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

B. Y. Chin, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM

First BALANC is called to balance the real matrix and isolate eigenvalues whenever possible. Then ORTHES is called to reduce the submatrix specified by BALANC to upper Hessenberg form by orthogonal similarity transformations. HQR is then called to find the eigenvalues of a real upper Hessenberg matrix by the QR method. INVIT is called to find those eigenvectors of a real upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration. ORTBAK and BALBAK then form the desired eigenvectors by back transforming those of the corresponding upper Hessenberg matrix determined by ORTHES. These eigenvectors are then normalized so that the component of largest magnitude is 1.

Finally, the eigenvalues are arranged in ascending order of magnitude; those with equal magnitude are ordered by increasing argument between 0 to 2π . The selected eigenvectors are arranged to correspond to the ordered eigenvalues.

SPACE REQUIRED

$5332_8 = 2778_{10}$

ACCURACY

The eigenvalues and eigenvectors will usually be determined to within an absolute error of 10^{-14} .

TIMING

The approximate execution time for a matrix of order 20 on the CDC 7600 is as follows:

All values, all vectors = 590 milliseconds
 All values, no vectors = 55 milliseconds

The execution time necessary to obtain all the eigenvalues is roughly proportional to N^2 . The execution time necessary to obtain each eigenvector is proportional to N .

PORTABILITY

There is a machine dependent constant defined and used in the subroutines HQR and INVIT:

```
"MACHEP=2.**(-47)"
```

ULIBER, a system resident routine, is used for printing error messages. A FORTRAN version of this routine is on ULIB.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, CABS, ATAN2

SUBROUTINE EIGSFM (NDIM,N,AR,TYPE,IDEF,EPS1,M,KF,W,Z,IERR,JERR,WORK)

**DIMENSION OF
ARGUMENTS**

AR(NDIM,N),W(M),Z(NDIM,M),WORK(10*N)

LATEST REVISION

May 1974

PURPOSE

This subroutine finds the algebraically smallest or largest M eigenvalues of a real symmetric matrix. Optionally it will also compute the eigenvectors corresponding to those eigenvalues.

Note: This subroutine is designed to find a few eigenvalues and corresponding eigenvectors. If all or almost all of the eigenvalues and eigenvectors are desired subroutine EIGSFS is recommended.

ACCESS CARDS

*FORTRAN,S=ULIB,N=EIGSFM
*COSY

USAGE

CALL EIGSFM (NDIM,N,AR,TYPE,IDEF,EPS1,M,KF,W,Z,IERR,JERR,WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension of the two dimensional arrays AR and Z in the calling program.

N

The order of the matrix.

AR

An array containing the matrix. Because of symmetry only the diagonal and the lower triangular part need be supplied. The row dimension of AR in the calling program must be equal to NDIM.

TYPE

A logical variable set .TRUE. if the smallest eigenvalues are to be found and set .FALSE. if the largest eigenvalues are to be found.

IDEF

An integer variable set to 1 if the matrix is known to be positive definite, set to -1 if the matrix is known to be negative definite, and set to 0 otherwise.

EPS1

An absolute error bound used in computing the eigenvalues. If EPS1 is specified to be non-positive, the routine will compute a default value for EPS1. If eigenvectors are desired, it is suggested that EPS1 be set non-positive.

M

An integer variable set equal to the number of extreme eigenvalues desired.

KF

A flag which is set to indicate if the corresponding eigenvectors are desired.

= 1 all eigenvectors for those M eigenvalues are desired
= 0 no eigenvectors are desired

WORK

An array which must be provided for workspace. WORK must be dimensioned at least $10*N$ if the eigenvectors are desired. If no eigenvectors are desired, it need be dimensioned only $5*N$.

On Output

W

An array of dimension M which contains the M extreme eigenvalues of the matrix. If the smallest eigenvalues have been found, they are arranged in ascending order in W. If the largest eigenvalues have been found, they are arranged in descending order in W.

Z

An array, with row (first) dimension NDIM and column dimension at least M, containing the computed eigenvectors stored (column-wise) to correspond with the W array. The eigenvectors are normalized so that the component of largest magnitude is 1.

IERR

- = 0 no error has occurred
- = $6*N+1$ If IDEF is set to 1 and TYPE is set .TRUE. when the input matrix is not positive definite, or if IDEF is set to -1 and TYPE is set .FALSE. when the matrix is not negative definite. No eigenvalues are computed. An error message will be printed out.
- = $5*N+K$ If successive iterates to the K^{th} eigenvalue are not strictly monotone increasing. The sum of all the shifts up to this point is taken as the eigenvalue and the program proceeds to the next eigenvalue calculation. If this failure occurs for more than one eigenvalue, the last occurrence is recorded in IERR. An error message will be printed out.

JERR

- = 0 no error has occurred
- = -R if the iteration for the eigenvector corresponding to the R^{th} eigenvalues fails to converge in 5 iterations. The R^{th} column of Z is set to zero and an error message will be printed out.

ENTRY POINTS

EIGSFM, TRED1, RATQR, TINVT, TRBAK1

SPECIAL CONDITIONS

All the algebraically smallest or largest M eigenvalues are sought. If IDEF is set to 1 and TYPE is set .TRUE. when the input matrix is not positive definite, or if IDEF is set to -1 and TYPE is set .FALSE. when the matrix is not negative definite, no eigenvalues are computed. An error message will be printed out. Control is returned to the calling program.

If successive iterates to the k^{th} eigenvalue are not strictly monotone increasing, the sum of all the shifts up to this point is taken as the eigenvalue and the program proceeds to the next eigenvalue calculation. If this failure occurs an error message will be printed out.

If KF = 1, eigenvectors will be sought for all the above eigenvalues. If the iteration for a vector fails to converge, it will be set to zero. An error message will be printed out and computation continued.

COMMON BLOCKS

None

I/O

See "SPECIAL CONDITIONS".

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

B. Y. Chin, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM

The given real, symmetric matrix is reduced to a symmetric, tridiagonal matrix by the subroutine TRED1 using orthogonal similarity transformations. The subroutine RATQR then finds the algebraically smallest or largest M eigenvalues according to specification, using rational QR method with Newton correction. If requested, the eigenvectors of the tridiagonal symmetric matrix corresponding to these eigenvalues are found by the subroutine TINVIT using inverse iteration. These eigenvectors are then transformed by the subroutine TRBAK1 to form the eigenvectors of the original matrix. The eigenvectors are then normalized so that the component of largest magnitude is equal to 1.

SPACE REQUIRED $2462_8 = 1280_{10}$ **ACCURACY**

Eigenvalues will be computed to within an absolute error given by EPS1 when this parameter is specified to be positive. When EPS1 is specified non-positive, the routine computes a value of EPS1 adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of $2.^{-47}$ in the matrix elements. A more detailed discussion of the use of EPS1 may be found in the EISPACK write-up.

TIMING

The approximate running time on the NCAR CDC 7600 is $0.1N^2$ milliseconds to find all the eigenvalues and corresponding eigenvectors.

PORTABILITY

There is a machine dependent constant defined and used in the subroutines RATQR and TINVIT:

```
"MACHEP=2.**-47"
```

ULIBER, a system resident routine, is used for printing error messages. A FORTRAN version of this routine is on ULIB.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

EIGSFS

SUBROUTINE EIGSFS (NDIM,N,AR,WR,ZR,EPS1,XLB,UB,NMX,KF,IERR,WORK)

**DIMENSION OF
ARGUMENTS**

AR(NDIM,N),WR(NMX),ZR(NDIM,NMX),WORK(N*(2*NDIM+7)+NMX)

LATEST REVISION

February 1973

PURPOSE

This subroutine finds all the eigenvalues of a given real, symmetric matrix which lie in a given interval. Optionally it will also compute the eigenvectors corresponding to those eigenvalues.

ACCESS CARDS

*FORTRAN,S=ULIB,N=EIGSFS
*COSY

USAGE

CALL EIGSFS (NDIM,N,AR,WR,ZR,EPS1,XLB,UB,NMX,KF,IERR,WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension of the two dimensional arrays AR and ZR in the calling program.

N

The order of the matrix.

AR

An array containing the matrix. Because of symmetry only the diagonal and the lower triangular part need be supplied. The row dimension of AR in the calling program must be equal to NDIM.

EPS1

An absolute error bound used in computing the eigenvalues. If EPS1 is specified to be non-positive, the routine will compute a default value for EPS1. If eigenvectors are desired, it is suggested that EPS1 be set non-positive.

XLB, UB

These real variables define the interval to be searched for eigenvalues. XLB must be less than UB.

NMX

An integer which should be set to an upper bound for the number of eigenvalues in the interval.

Warning: If more than NMX eigenvalues are computed in the interval, execution stops and control is returned to the calling program with NMX set equal to the actual number of eigenvalues found and no eigenvalues returned in the WR array.

KF

A flag which is set to indicate if the corresponding eigenvectors are desired.

= 0 no eigenvectors are desired.

= 1 all the eigenvectors are desired for those eigenvalues found in the specified intervals.

WORK

An array which must be provided for workspace. WORK must be dimensioned at least $(N*(2*NDIM+7)+NMX)$ if the eigenvectors are desired. If no eigenvectors are desired, it need be dimensioned only $(N*(2*NDIM+4)+NMX)$.

On Output

WR

An array of dimension NMX which contains the eigenvalues arranged in ascending order.

ZR

An array containing the eigenvectors (stored column-wise) corresponding to the WR array. The row dimension of ZR in the calling program must be equal to NDIM. Each vector is normalized so that its component of largest magnitude is one. If KF=0, ZR may be a dummy variable.

EPS1

If EPS1 was originally non-positive, it will be set to its last default value.

On Output
(continued)

IERR

An error flag.

- = 0, no error has occurred.
- = 1, $XLB \geq UB$
- = $3*N+1$, too many eigenvalues have been found in the specified interval. An error message will be printed out.
- = -R, if the iteration for the eigenvector corresponding to the R^{th} eigenvalue fails to converge in 5 iterations. The R^{th} column of ZR is set to zero and an error message is printed out.

ENTRY POINTS

EIGHFS, TRED1, BISECT, TINVIT, TRBAKL

SPECIAL CONDITIONS

All the eigenvalues in a specified interval (XLB, UB) are sought. If the number of eigenvalues in that interval exceeds a maximum number, NMX, an error message will be printed out and no eigenvalues will be returned to the calling program. Control is returned to the calling program.

If $KF \neq 0$, eigenvectors will be sought for all the above eigenvalues. If the iteration for a vector fails to converge, it will be set to zero. An error message will be printed out and computation continued.

COMMON BLOCKS

None

I/O

See "Special Conditions"

PRECISION Single

REQUIRED ULIB
ROUTINES None

SPECIALIST Nancy Werner, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM The given real, symmetric matrix is reduced to a symmetric, tridiagonal matrix by the subroutine TRED1 using orthogonal similarity transformations. The subroutine BISECT then finds those eigenvalues of this matrix which lie in a specified interval, using bisection.

If requested, the eigenvectors of the tridiagonal matrix corresponding to these eigenvalues are found by the subroutine TINVIT using inverse iteration. These eigenvectors are then transformed by the subroutine TRBAK1 to form the eigenvectors of the original matrix. The eigenvectors are then normalized so that the component of largest magnitude is equal to one.

SPACE REQUIRED

2603₈ = 1411₁₀

ACCURACY

Eigenvalues will be computed to within an absolute error given by EPS1 when this parameter is specified to be positive. When EPS1 is specified non-positive, the routine computes a value of EPS1 adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of $2. \times 10^{-47}$ in the matrix elements. A more detailed discussion of the use of EPS1 may be found in the EISPACK write-up.

TIMING

The approximate running time on the NCAR CDC 7600 is $0.6 \times 10^{-3} N^3$ milliseconds to reduce the given matrix to tridiagonal form. To find all the eigenvalues and corresponding eigenvectors requires $0.02 N^2$ milliseconds.

PORTABILITY

There is a machine dependent constant defined and used in the subroutines BISECT and TINVIT:

```
"MACHEP=2.**(-47)"
```

ULIBER, a system resident routine, is used for printing error messages. A FORTRAN version of this routine is on ULIB.

REQUIRED RESIDENT
ROUTINES

ULIBER

SUBROUTINE EIGSTM (NDIM,N,EPS1,D,E,TYPE,IDEF,M,KF,W,Z,IERR,JERR,WORK)

**DIMENSION OF
ARGUMENTS**

D(N),E(N),W(M),Z(NDIM,M),WORK(8*N)

LATEST REVISION

May 1974

PURPOSE

This subroutine finds the algebraically smallest or largest M eigenvalues of a real symmetric tridiagonal matrix. Optionally it will also compute the eigenvectors corresponding to those eigenvalues.

Note: This subroutine is designed to find a few eigenvalues and corresponding eigenvectors. If all or almost all of the eigenvalues and eigenvectors are desired subroutine EIGSTS is recommended.

ACCESS CARDS

*FORTRAN,S=ULIB,N=EIGSTM
*COSY

USAGE

CALL EIGSTM (NDIM,N,EPS1,D,E,TYPE,IDEF,M,KF,W,Z,IERR,JERR,WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension of the two dimensional array Z in the calling program.

N

The order of the matrix.

EPS1

An absolute error bound used in computing the eigenvalues. If EPS1 is specified to be non-positive, the routine will compute a default value for EPS1. If eigenvectors are desired, it is suggested that EPS1 be set non-positive.

D

A real array of dimension N containing the diagonal elements of the real symmetric tridiagonal matrix.

E

A real array of dimension N containing, in its last N-1 positions, the subdiagonal elements of the real symmetric tridiagonal matrix. E(1) is arbitrary.

TYPE

A logical variable set .TRUE. if the smallest eigenvalues are to be found and set .FALSE. if the largest eigenvalues are to be found.

IDEF

An integer variable set to 1 if the matrix is known to be positive definite, set to -1 if the matrix is known to be negative definite, and set to 0 otherwise.

M

An integer variable set equal to the number of extreme eigenvalues desired.

KF

A flag which is set to indicate if the corresponding eigenvectors are desired.

- = 1 all eigenvectors for those M eigenvalues are desired
- = 0 no eigenvectors are desired

WORK

An array which must be provided for workspace. WORK must be dimensioned at least $8*N$ if the eigenvectors are desired. If no eigenvectors are desired, it need be dimensioned only $3*N$.

On Output

W

An array of dimension M which contains the M extreme eigenvalue of the matrix. If the smallest eigenvalues have been found, they are arranged in ascending order in W. If the largest eigenvalues have been found, they are arranged in descending order in W.

Z

An array, with row (first) dimension NDIM and column dimension at least M, containing the computed eigenvectors stored (column-wise) to correspond with the W array. The eigenvectors are normalized so that the component of largest magnitude is one.

IERR

- = 0 no error has occurred
- = $6*N+1$ if IDEF is set to 1 and TYPE is set .TRUE. when the input matrix is not positive definite, or if IDEF is set to -1 and TYPE is set .FALSE. when the matrix is not negative definite, no eigenvalues are computed. An error message is printed out.
- = $5*N+K$ if successive iterates to the K^{th} eigenvalue are not strictly monotone increasing, the sum of all the shifts up to this point is taken as the eigenvalue and the program proceeds to the next eigenvalue calculation. If this failure occurs for more than one eigenvalue, the last occurrence is recorded in IERR. An error message will be printed out.

On Output
(continued)

JERR
 = 0 no error has occurred
 = -R if the iteration for the eigenvector corresponding to the R^{th} eigenvalues fails to converge in 5 iterations. The R^{th} column of Z is set to zero and an error message will be printed out.

ENTRY POINTS

EIGSTM, RATQR, TINVIT

SPECIAL CONDITIONS

All the algebraically smallest or largest M eigenvalues are sought. If IDEF is set to 1 and TYPE is set .TRUE. when the input matrix is not positive definite, or if IDEF is set to -1 and TYPE is set .FALSE. when the matrix is not negative definite, no eigenvalues are computed. An error message will be printed out. Control is returned to the calling program.

If successive iterates to the K^{th} eigenvalue are not strictly monotone increasing, the sum of all the shifts up to this point is taken as the eigenvalue and the program proceeds to the next eigenvalue calculation. If this failure occurs an error message will be printed out.

If KF = 1, eigenvectors will be sought for all the above eigenvalues. If the iteration for a vector fails to converge, it will be set to zero. An error message will be printed out and computation continued.

COMMON BLOCKS

None

I/O

See "SPECIAL CONDITIONS".

REQUIRED ULIB ROUTINES None

SPECIALIST B. Y. Chin, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM The algebraically smallest or largest M eigenvalues of a given real, symmetric, tridiagonal matrix are computed by subroutine RATQR, using the rational QR method with Newton corrections. If requested, the eigenvectors of the given matrix corresponding to these eigenvalues are found by the subroutine TINVIT. The eigenvectors are then normalized so that the component of largest magnitude is equal to one.

SPACE REQUIRED $1644_8 = 932_{10}$

ACCURACY Eigenvalues will be computed to within an absolute error given by EPS1 when this parameter is specified to be positive. When EPS1 is specified non-positive, the routine computes a value of EPS1 adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of $2.***(-47)$ in the matrix elements. A more detailed discussion of the use of EPS1 may be found in the EISPACK write-up.

TIMING

The approximate running time on the NCAR CDC 7600 is $0.065N^2$ milliseconds to find all the eigenvalues and corresponding eigenvectors.

PORTABILITY

There is a machine dependent constant defined and used in the subroutines RATQR and TINVIT:

```
"MACHEP=2.**(-47)"
```

ULIBER, a system resident routine, is used for printing error messages. A FORTRAN version of this routine is on ULIB.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

EIGSTS

SUBROUTINE EIGSTS (NDIM,N, EPS1,D,E,RLB,UB,MM,M,W,Z,KF,IERR,WORK)

**DIMENSION OF
ARGUMENTS**

D(N),E(N),W(MM),Z(NDIM,MM),WORK(7*N+MM)

LATEST REVISION

May 1974

PURPOSE

This subroutine finds all the eigenvalues of a given real, symmetric, tridiagonal matrix which lie in a given interval. Optionally it will also compute the eigenvectors corresponding to those eigenvalues.

ACCESS CARDS

*FORTRAN,S=ULIB,N=EIGSTS
*COSY

USAGE

CALL EIGSTS (NDIM,N, EPS1,D,E,RLB,UB,MM,M,W,Z,KF,IERR,WORK)

ARGUMENTS

On Input

NDIM

The row (first) dimension of the two-dimensional array Z in the calling program.

On Input
(continued)

N

The order of the matrix.

EPS1

An absolute error bound used in computing the eigenvalues. If EPS1 is specified to be non-positive, the routine will compute a default value for EPS1. If eigenvectors are desired, it is suggested that EPS1 be set non-positive.

D

A real array of dimension N containing the diagonal elements of the symmetric tridiagonal matrix.

E

A real array of dimension N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

RLB, UB

These real variables define the interval to be searched for eigenvalues. RLB must be less than UB.

MM

An integer specifying an upper bound on the number of eigenvalues in the interval (RLB, UB). If more than MM eigenvalues are found, execution ceases and control is returned to the calling program with M set equal to the actual number of eigenvalues found and with no eigenvalues specified in the output vector.

KF

A flag which is set to indicate if the corresponding eigenvectors are desired.

= 0 no eigenvectors are desired

= 1 all eigenvectors for those eigenvalues found in the specified interval are desired

WORK

An array which must be provided for workspace. WORK must be dimensioned at least $(7*N+MM)$ if the eigenvectors are desired. If no eigenvectors are desired, it need be dimensioned only $(3*N+MM)$.

On Output

M

An integer variable set equal to the number of eigenvalues determined to lie in the interval (RLB,UB).

W

An array of dimension MM which contains the M eigenvalues arranged in ascending order.

Z

An array, with row (first) dimension NDIM and column dimension at least MM, containing the computed eigenvectors stored (column-wise) to correspond with the W array. The eigenvectors are normalized so that the component of largest magnitude is one.

EPS1

If EPS1 was originally non-positive, it will be set to its last default value.

IERR

An error flag.

- = 0 no error has occurred
- = 1 $RLB \geq UB$
- = $3*N+1$ too many eigenvalues have been found in the specified interval. An error message will be printed out.
- = -R if the iteration for the eigenvector corresponding to the R^{th} eigenvalue fails to converge in 5 iterations. The R^{th} column of Z is set to zero and an error message will be printed out.

ENTRY POINTS

EIGSTS, BISECT, TINVIT

SPECIAL CONDITIONS

All the eigenvalues in a specified interval (RLB,UB) are sought. If the number of eigenvalues in that interval exceeds a maximum number, MM, an error message will be printed out and no eigenvalues will be returned to the calling program. Control is returned to the calling program.

SPECIAL CONDITIONS
(continued)

If $KF = 1$, eigenvectors will be sought for all the above eigenvalues. If the iteration for a vector fails to converge, the corresponding eigenvector will be set to zero. An error message will be printed out and computation continued.

COMMON BLOCKS

None

I/O

See "SPECIAL CONDITIONS".

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

B. Y. Chin, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

This subroutine is principally composed of subroutines obtained from EISPACK, the eigenvalue package distributed by Argonne National Laboratory. Information on this package is available on request.

ALGORITHM

All the eigenvalues in a specified interval of a given real, symmetric, tridiagonal matrix are computed by subroutine BISECT, using bisection. If requested, the eigenvectors of the given real, symmetric, tridiagonal matrix corresponding to these eigenvalues are found by the subroutine TINVIT. The eigenvectors are then normalized so that the component of largest magnitude is equal to one.

SPACE REQUIRED

$1743_8 = 995_{10}$

ACCURACY

Eigenvalues will be computed to within an absolute error given by EPS1 when this parameter is specified to be positive. When EPS1 is specified non-positive, the routine computes a value of EPS1 adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of $2. \cdot 10^{-47}$ in the matrix elements. A more detailed discussion of the use of EPS1 may be found in the EISPACK write-up.

TIMING

The approximate running time on the NCAR CDC 7600 is $.4N^2$ milliseconds to find all the eigenvalues and corresponding eigenvectors.

PORTABILITY

There is a machine dependent constant defined and used in the subroutines BISECT and TINVIT:

```
"MACHEP=2. \cdot 10^{-47}"
```

ULIBER, a system resident routine, is used for printing error messages. A FORTRAN version of this routine is on ULIB.

3.EIGSTS. 6

**REQUIRED RESIDENT
ROUTINES**

ULIBER

OVERDETERMINED LINEAR SYSTEM SOLVER

LATEST REVISION October 1973

PURPOSE The package HSHSLV calculates the solution vector for a real overdetermined linear system. HSHSLV contains three subroutines; HDEC, SOLVEH and HSITIM. HDEC decomposes the overdetermined matrix using Householder transformations. SOLVEH calculates the solution vector. HSITIM performs iterative improvement on the calculated solution vector.

ACCESS CARDS *FORTRAN,S=ULIB,N=HSHSLV
 *COSY

SPACE REQUIRED $1214_8 = 652_{10}$

USAGE CALL HDEC(A,AH,VC,MA,MAH,M,N,IP,TEMP,ISW)
 CALL SOLVEH(AH,VC,B,TEMP,X,MAH,M,N,IP,ISW)

If the original matrix A has not been destroyed, then

CALL HSITIM(A,AH,VC,X,B,TEMP,MA,MAH,M,N,ITS,IP)

ARGUMENTS

On Input
for HDEC

A

Is a real input two-dimensional variable with row dimension MA and column dimension N. On input, A contains the overdetermined matrix of the linear system.

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

MAH

Is an integer input variable set equal to the row dimension of the matrix AH as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the actual row dimension of the matrix A used in HDEC.

N

Is an integer input variable set equal to the column dimension of the matrix A.

TEMP

Is a real vector variable with dimension at least N. It is used internally by the subroutine for working storage.

On Output
for HDEC

AH

Is a real two-dimensional variable with row dimension MAH and column dimension N. On output, AH contains the Householder decomposition of the matrix A.

If A is entered twice in the parameter list, replacing AH, then, on output, A will contain the elements of AH, and the original A will be destroyed.

VC

Is a real vector variable with dimension N. On output, VC contains additional information about the Householder decomposition of the matrix A.

IP

Is an integer vector variable with dimension N. On output, IP contains information about the column permutations of the matrix A.

ISW

Is an integer error flag.

= 0 if the matrix A is non-singular.

= if the matrix A is singular. The integer K is the number of columns which have been decomposed.

On Input
for SOLVEH

AH

Is the real two-dimensional variable calculated in HDEC.

VC

Is the real vector variable calculated in HDEC.

B

Is a real vector variable with dimension M. On input, B contains the elements of the right hand side of the linear system.

TEMP

Is a real vector variable with dimension at least M. It is used internally for working storage.

MAH

Is an integer variable set equal to the row dimension of AH as declared in the dimension statement of the calling program.

M

Is an integer variable set equal to the actual row dimension of AH used in SOLVEH.

On Input
for SOLVEH
(continued)

N
Is an integer variable set equal to the column dimension of AH.

IP
Is an integer vector variable determined in HDEC.

On Output
for SOLVEH

X
Is a real vector variable with dimension N. On output, X contains the calculated solution vector.

ISW
Is an integer error flag.
= 1 if AH has a zero diagonal element.
= 0 otherwise.

On Input
for HSITIM

A
Is the matrix described in input for HDEC.

AH
Is the matrix described in output for HDEC.

VC
Is the vector described in output for HDEC.

X
Is the vector described in output for SOLVEH.

B
Is the vector described in input for SOLVEH.

TEMP
Is a real vector variable with dimension at least $N + M$. It is used internally for working storage.

MA

Is the row dimension of A declared in the dimension statement of calling program.

MAH

Is the row dimension of AH declared in the dimension statement of calling program.

M

Is the row dimension of A actually used.

N

Is the column dimension of A.

ITS

Is an integer variable set equal to the maximum number of iterations for iterative improvement.

IP

Is the vector variable described in the output for HDEC.

On Output
for HSITIM

X

Is the vector variable containing the improved solution vector.

ENTRY POINTS

HDEC, SOLVEH, HSITIM

COMMON BLOCKS

None

I/O

All I/O messages are printed using subroutine ULIBER.

For HDEC, if the matrix A is singular, the message

MATRIX SINGULAR IN SUBROUTINE HDEC

is printed.

For SOLVEH, if AH has a zero diagonal element, the message,

MATRIX SINGULAR IN SUBROUTINE SOLVEH

is printed.

For HSITIM, if the number of iterations reaches the value of the variable ITS, the subroutine prints the message,

EXCEEDS ITERATIONS IN HSITIM.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

ULIBER

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Written August 1972, Standardized October 1973.

ALGORITHM

The Householder transformations decompose the general matrix A into the product of an upper triangular matrix U and an orthogonal matrix Q , where Q is the product of N orthogonal transformations $Q(N), \dots, Q(1)$, with each $Q(I) = \text{IDENT} - B * W(I) * W(I)^T$; B a scalar and $W(I)$ a vector with zeros in the first $I - 1$ components, and the elements of the I^{th} column of the matrix A as the $I+1, I+2, \dots, M$ components. Each $Q(I)$ introduces zeros into the strictly subdiagonal elements of the I^{th} column of A .

Beginning with $I = 1$, the column norms, S , of A are calculated. A permutation of the I^{th} column and the column of maximum norm is performed.

Next, the I^{th} element of $W(I)$ is calculated, which equals $A(I, I) + \text{or} - S(I)$ (norm of the column). The sign chosen is the same as the sign of $A(I, I)$. This value is stored in $VC(I)$.

The previous value of $A(I, I)$ is replaced by $S(I) * \text{SIGN}(A(I, I))$. The transformation $Q(I) = \text{IDENT} - B * W(I) * W(I)^T$ is applied to the $I + 1$ through N columns of A . The process is repeated until N orthogonal transformations have been formed and applied to A .

To solve, each of the orthogonal matrices, $Q(1), \dots, Q(N)$, is applied to the vector B and stored in TEMP . Back substitution is applied to the upper triangular system $U * X = Q^T * B$ solving for X . Finally the vector X is permuted as indicated by the vector IP .

ALGORITHM

(continued)

For iterative improvement the residual vector $B - A*X$ is calculated in double precision and stored in the vector TEMP. Using subroutine SOLVEH, the linear system $A*DX = TEMP$ is solved for DX.

If the ratio of the maximum element of DX to the maximum element of X is less than machine precision, the subroutine terminates. Otherwise, the correction vector DX is added to X and the process is repeated from the beginning. If the number of iterations reaches the value of the variables ITS, the subroutine terminates.

ACCURACY

The accuracy of the solution depends on the conditioning of the matrix A and the error in the right hand side B.

TIMING

The timing is proportional to the quantity MN^2 . For $M = 60$ and $N = 40$, the time is 83 milliseconds on the CDC 7600.

PORTABILITY

There are no machine dependent constants.

**REQUIRED RESIDENT
ROUTINES**

SQRT

INVMTX

SUBROUTINE INVMTX (A,NA,V,NV,N,D,IP,IER)

DIMENSION OF ARGUMENTS A(NA,N),V(NV,N),IP(2*N)

LATEST REVISION April 1974

PURPOSE INVMTX calculates the inverse of the N×N input matrix A using Gaussian elimination with full pivoting.

ACCESS CARDS *FORTRAN,S=ULIB,N=INVMTX
 *COSY

USAGE CALL INVMTX (A,NA,V,NV,N,D,IP,IER)

ARGUMENTS

On Input A

A two-dimensional variable with row dimension NA and column dimension N. On input, A contains the elements of the matrix to be inverted.

On Input
(continued)

NA

An integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

NV

An integer input variable set equal to the row dimension of V as declared in the dimension statement of the calling program.

N

An integer input variable set equal to the column dimension of A.

IP

An integer array used internally for working storage. It must have dimension at least $2*N$.

On Output

V

A two-dimensional variable with row dimension NV and column dimension N. On output, V contains the inverse of A.

If A is entered twice in the parameter list, replacing V, then on output the array A will contain the inverse matrix, and the original A will be destroyed.

D

A real variable which on output contains the determinant of A.

IER

An integer error flag.

= 33 if the matrix A is singular
= 0 otherwise

ENTRY POINTS

INVMIX

COMMON BLOCKS None

I/O The message is printed using ULIBER. If the matrix A has a zero pivot element (i.e., A is singular), the message

MATRIX SINGULAR IN INVMTX

is printed.

PRECISION Single

**REQUIRED ULIB
ROUTINES** None

SPECIALIST Revised for NSSL by Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Standardized April, 1974.

ALGORITHM The subroutine INVMTX solves the matrix equation

$A*V = I$

for the matrix V (i.e., A^{-1}), where I is the identity matrix.

ALGORITHM
(continued)

The method used is Gaussian elimination with full pivoting. The matrix A is decomposed into the product of a lower triangular matrix and an upper triangular matrix

$$A = L*U$$

thus

$$\begin{aligned} A*V &= I \\ (L*U)*V &= I \\ U*V &= L^{-1} \end{aligned}$$

This upper triangular matrix equation is solved for the columns of V using back substitution.

If at any point, after pivoting is done, a pivot element is equal to zero, the matrix A is declared singular and the subroutine terminates.

SPACE REQUIRED

$$352_8 = 234_{10}$$

ACCURACY

INVMIX may fail to yield accurate solutions if the matrix A is sufficiently ill-conditioned. For a discussion of the condition of a matrix see Wilkinson, "The Algebraic Eigenvalue Problem."

TIMING

The time required by INVMIX is proportional to the quantity N^3 . On the CDC 7600 with $N=20$, INVMIX takes 16 milliseconds.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

LINEQSV**LATEST REVISION**

October 1973

PURPOSE

LINEQSV is a package of three subroutines--DECOMP, SOLVE and IMPRUV--which may be used to solve a system of linear equations by Gaussian elimination with partial pivoting and (optionally) with iterative improvement.

ACCESS CARDS

*FORTRAN,S=ULIB,N=LINEQSV
*COSY

USAGE

To solve a system of equations, $Ax=b$, the user first calls DECOMP by

```
CALL DECOMP (N,NDIM,A,LU,D,IPS,IER)
```

which decomposes A into the product of a unit lower triangular matrix and an upper triangular matrix. After checking D to insure that the determinant of A is not zero, the user then follows this call with a call to SOLVE by

```
CALL SOLVE (N,NDIM,LU,B,IPS,X)
```

3.LINEQSV.2

USAGE

(continued)

which uses the decomposition to find the solution x . If desired, the user may then try to improve the accuracy of the solution by calling IMPRUV by

```
CALL IMPRUV (N,NDIM,A,LU,B,IPS,X,R,DIGITS,IER)
```

If the user wants to solve several linear systems of equations with the same coefficient matrix A , only one call to DECOMP is necessary. This is then followed by a call to SOLVE for each different b vector.

Each of the above routines is written up in detail below.

ENTRY POINTS

DECOMP, SOLVE, IMPRUV

SPACE REQUIRED

$723_8 = 467_{10}$ locations

SUBROUTINE DECOMP (N,NDIM,A,LU,D,IPS,IER)

DIMENSION OF ARGUMENTS A(NDIM,N),LU(NDIM,N),IPS(N)

PURPOSE Decompose matrix into the product of upper and unit lower triangular matrices using Gaussian elimination with partial pivoting.

USAGE CALL DECOMP (N,NDIM,A,LU,D,IPS,IER)

ARGUMENTS

On Input N
 Order of matrix A.

 NDIM
 Declared first (row) dimension of A and LU in the calling program.

 A
 Matrix to be decomposed.

On Output LU
 Lower and upper triangular matrices of A, where
 $LU(I,J)$, for $I \leq J$,
 is the upper triangular matrix with row interchanges completed, and

On Output
(continued)

LU(I,J), for $I > J$,

is the negative of lower triangular matrix excluding diagonal elements, but with row interchanges partially completed.

If matrix A need not be saved, then the matrix A may be used as the decomposition matrix LU in the call, i.e.,

CALL DECOMP (N,NDIM,A,A,D,IPS,IER)

Note: If subroutine IMPRUV will be used, A must be retained.

D

Determinant of A.

IPS

Vector (of dimension at least N) storing row permutations resulting from pivoting, where $IPS(K) = K$ th pivot row after row interchanges from previous pivots.

IER

Error flag used by standard error message routine, ULIBER, and returned to user.

= 0, no error.

= 32, determinant of A equals zero, hence decomposition is impossible. A message is printed.

NOTE

An NCAR system routine, LOC, is used to check equivalence of arguments A and LU by checking their addresses to eliminate unnecessary transfer of A to LU when possible. ULIBER is called to output error messages.

COMMON BLOCKS

None

I/O

An NCAR resident routine, ULIBER, is used to output error messages.

PRECISION Single

**REQUIRED ULIB
ROUTINES** None

SPECIALIST R. K. Sato, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Transcribed from a paper by C. B. Moler, "Linear Equation Solver," Comm. ACM 15 (April, 1972), 74.

ALGORITHM Uses Gaussian elimination with partial pivoting with the computations arranged so that the inner loops vary column indices rather than row indices. The row interchanges from pivoting are completed in U but only partially completed in L.

ACCURACY The accuracy is dependent upon the condition of the matrix and decreases rapidly as the matrix becomes more ill-conditioned.

TIMING The time required for a 30×30 system on the NCAR CDC 7600 is approximately 9.5 milliseconds. The time is proportional to n^3 .

PORTABILITY

Two NCAR routines, LOC and ULIBER, are called. (See "Note".)
A FORTRAN version of ULIBER is on a ULIB file.

**REQUIRED RESIDENT
ROUTINES**

LOC, ULIBER (See "Note".)

SUBROUTINE SOLVE (N,NDIM,LU,B,IPS,X)

DIMENSION OF ARGUMENTS LU(NDIM,N),B(N),IPS(N),X(N)

PURPOSE Solves for vector x of unknowns of the linear system $Ax=b$ using the LU decomposition of A from subroutine DECOMP.

USAGE CALL SOLVE (N,NDIM,LU,B,IPS,X)

ARGUMENTS**On Input**

N
Order of matrix A (number of equations).

NDIM
Declared first (row) dimension of LU in the calling program.

LU
LU decomposition matrix from subroutine DECOMP.

B
Right hand side vector (of dimension at least N) of constants.

IPS
Vector (of dimension at least N) from subroutine DECOMP storing row permutations resulting from pivoting, where $IPS(K) = K$ th pivot row after row interchanges from previous pivots.

On Output

X

Solution vector (of dimension at least N). If vector B need not be saved, then the vector B may be used as the solution vector in the call, i.e.,

CALL SOLVE (N,NDIM,LU,B,IPS,B)

Note: If subroutine IMPRUV will be used, B must be retained.

NOTE

An NCAR system routine, LOC, is used to check equivalence of arguments B and X by checking their addresses to eliminate unnecessary transfer of B to X when possible.

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

R. K. Sato, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Transcribed from a paper by C. B. Moler, "Linear Equation Solver," Comm. ACM 15 (April, 1972), 74.

ALGORITHM

Using the LU decomposition from DECOMP, the original system $(Ax=b)$ is solved by solving the equivalent two triangular systems

$$Lg=b$$

$$Ux=g$$

During the solution of the first system, the row interchanges which were only partially completed in subroutine DECOMP are completed.

ACCURACY

The accuracy of the solution depends upon the LU decomposition from DECOMP. The accuracy of the decomposition depends upon the condition of the coefficient matrix and decreases rapidly as the matrix becomes more ill-conditioned.

TIMING

The time required for a 30×30 system on the NCAR CDC 7600 is approximately 0.8 milliseconds. The time is proportional to n^2 .

PORTABILITY

An NCAR system routine, LOC, is called. (See "Note".)

**REQUIRED RESIDENT
ROUTINES**

LOC (See "Note".)

SUBROUTINE IMPRUV (N,NDIM,A,LU,B,IPS,X,R,DIGITS,IER)

DIMENSION OF ARGUMENTS

A(NDIM,N),LU(NDIM,N),B(N),IPS(N),X(N),R(N)

PURPOSE

Iterative improvement of the solution obtained from subroutine SOLVE.

USAGE

CALL IMPRUV (N,NDIM,A,LU,B,IPS,X,R,DIGITS,IER)

ARGUMENTS

On Input

N

Order of matrix A (number of equations).

NDIM

Declared first (row) dimensions of A and LU in the calling program.

A

Original coefficient matrix.

LU

LU decomposition matrix of A from subroutine DECOMP.

B

Original right hand side vector (of dimension at least N) of constants.

IPS

Vector (of dimension at least N) from subroutine DECOMP storing row permutations resulting from pivoting, where $IPS(K) = K$ th pivot row after row interchanges from previous pivots.

X
Solution vector (of dimension at least N) from
subroutine SOLVE.

R
Work array (of dimension at least N) used in the
computation of residuals.

On Output

X
Solution vector after iterative improvement.

DIGITS
Approximate number of correct digits in X.

IER
Error flag used by standard error message routine
ULIBER and returned to user.
= 0, no error.
= 1, no convergence in the iteration.

COMMON BLOCKS

None

I/O

An NCAR resident routine, ULIBER, is called to output the
error message.

PRECISION

Double precision is used in the calculation of the residuals.

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

R. K. Sato, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Transcribed from Forsythe and Moler, "Computer Solutions of Linear Algebraic Systems," 1967.

ALGORITHM

The improvement is based on the double precision computation of the residual vector, \bar{r}_i where

$$\bar{r}_i = \bar{b} - a\bar{x}_i \quad .$$

The system

$$A\bar{z} = \bar{r}_i$$

is solved for the vector \bar{z} . \bar{x}_{i+1} is defined as

$$\bar{x}_{i+1} = \bar{x}_i + \bar{z}$$

and is a solution of $A\bar{x} = \bar{b}$ since

$$\begin{aligned} A\bar{x}_{i+1} &= A(\bar{x}_i + \bar{z}) \\ &= A\bar{x}_i + A\bar{z} \\ &= A\bar{x}_i + \bar{r}_i \\ &= \bar{b} \end{aligned}$$

The improvement procedure is repeated on \bar{x}_{i+1} to obtain \bar{x}_{i+2} . The iteration is terminated when either

1. Number of iterations > ITMAX, or
2. $\max_j |x_{i+1,j} - x_{i,j}| < \text{EPS}(\max_j |x_{1,j}|)$ where $x_{i,j}$ is the j th component of the solution from the i th iteration.

ACCURACY

The accuracy of the solution is dependent upon the condition of the coefficient matrix. Systems with well-conditioned coefficient matrices will converge to results accurate to machine accuracy, but with ill-conditioned matrices there may be no improvement or convergence to a solution which is less accurate than the one returned from SOLVE.

TIMING

The time required for a 30×30 system on the NCAR CDC 7600 is approximately 2.5 milliseconds per iteration. The time is proportional to n^2 .

PORTABILITY

Two machine dependent constants, $EPS = 1.E - 15$ and $ITMAX = 30$, are defined in a data statement. A system routine, ULIBER, is called to output a message when there is no convergence. A FORTRAN version of ULIBER is on a ULIB file.

**REQUIRED RESIDENT
ROUTINES**

ALOG10, ULIBER

SUBROUTINE SUPRLS (I,ROWI,N,BI,A,NN,SOLN,ERR,IER)**DIMENSION OF
ARGUMENTS**

ROWI(N),A(NN),SOLN(N)

LATEST REVISION

June 1974

PURPOSE

To determine the least squares solution of a large over-determined linear system. Given the M by N matrix R ($M \geq N$) and the M-vector B, this routine calculates the N-vector X such that the Euclidean norm of the residue ($R*X-B$) is minimized. The subroutine accepts rows of the matrix one by one so that the entire matrix need not be stored at one time. This allows large problems to be solved without peripheral storage. The length of the rows is limited by the amount of scratch storage which can be set aside for use by the routine. There is no restriction on the number of rows.

ACCESS CARDS

*FORTRAN,S=ULIB,N=SUPRLS
*COSY

USAGE

The subroutine is called once for each row of the matrix. A final call returns the solution vector and the Euclidean norm of the residual. This following sequence would process the M by N matrix R and the right hand side M-vector B

```
DO 1 I = 1,M
DO 2 J = 1,N
```

Here set ROWI(J) to the (I,J) element of R

```
2 CONTINUE
```

Here set BI to the Ith component of B.

```
CALL SUPRLS(I,ROWI,N,BI,A,MN,SOLN,ERR,IER)
1 CONTINUE
CALL SUPRLS(0,ROWI,N,BI,A,MN,SOLN,ERR,IER)
```

ARGUMENTS**On Input**

I

The index of the row being entered. (I is 1 for the first call, increases by one for each call, and is M when the final row is entered). After the final row has been entered, SUPRLS is called with I = 0 to complete the reduction and solution.

ROWI

A vector which on the Ith call contains the N components of the Ith row of the matrix. The dimension of ROWI in calling program must be at least N.

N

The length of the rows of the matrix (i.e., the number of columns). $N \leq M$, where M is the number of rows.

BI

On the Ith call, BI contains the Ith element of the right hand side vector B.

A

A working array which must not be changed between the successive calls to SUPRLS. Dimension of A in calling program is NN.

NN

Length of scratch array A. NN must be at least $N*(N+5)/2+1$. For speed, NN should be as large as possible up to a maximum of $(N+1)*M$.

On Output

SOLN

The N-components of the solution vector one returned in this array after the final call to SUPRLS. Dimension of SOLN in calling program must be at least N.

ERR

The Euclidean norm of the residual is returned in ERR after the final call to SUPRLS.

IER

Error parameter.

Fatal errors.

- = 32 Insufficient scratch storage provided, must have $NN \geq N*(N+5)/2+1$
- = 33 Array has too few rows. Must have $M \geq N$.
- = 34 System is singular.
- = 35 Values of I not in sequence

ENTRY POINTS

SUPRLS

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O None

PRECISION Single

**REQUIRED ULIB
ROUTINES** ULIBER

SPECIALIST Cicely Ridley, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Written by A.K.Cline, NCAR, May 1972. Modified by Dave Fulker, NCAR, July 1973. Standardized and documented by Cicely Ridley, June 1974.

ALGORITHM Given the M by N matrix R ($M \geq N$) and the M-vector B, we wish to find an N-vector X such that

$$E = \|R^*X - B\|_2$$

is minimized. Since the Euclidean norm is invariant under orthogonal transformation, R and B may be premultiplied by any orthogonal matrix without changing the norm of the residual ($R^*X - B$). R is reduced to upper triangular form by premultiplying R and B by a sequence of Householder and rotation matrices. When the reduction is complete, the norm of the residual takes the form

$$E = \|T^*X - B_N\| + \|B_{M-N}\|$$

where T is an N by N upper triangular matrix, B_N is vector of first N components of B , B_{M-N} is vector of remaining $(M-N)$ components of B . E is minimized by taking X to be the solution of the system

$$T^*X = B_N \quad .$$

This triangular system is therefore solved to give the required least squares solution. The norm of the residual is then given by $\|B_{M-N}\|$.

At each phase of the reduction, as many rows as space permits are entered into the scratch area. Householder transformations are then used to zero out subdiagonal elements. Space is saved by eliminating storage for the zero subdiagonal terms. If there is room for only one new row, rotation rather than Householder matrices are used for greater speed. When all M rows have been entered, reduction is completed and the triangular system solved.

For greater detail see Hanson, R.J., and Lawson, C.L., 1969: Extensions and applications of the Householder algorithm for solving linear least squares problems. *Math. of Comp.* Vol.23, pp. 787-812.

SPACE REQUIRED

$1166_8 = 630_{10}$

ACCURACY

This will depend upon the size and condition of the matrix. Near machine accuracy may be expected for well conditioned systems of moderate size. If ill conditioning is suspect, a version using pivoting may be necessary.

TIMING

This depends not only upon the dimensions of the matrix, but also upon the amount of scratch storage available. For a 40 x 20 matrix and a maximum storage, the time taken on the CDC 7600 was 100 milliseconds. Reduction of scratch storage to a minimum nearly doubles this figure.

PORTABILITY

Portability standards are satisfied.

**REQUIRED RESIDENT
ROUTINES**

SQRT

PACKAGE SVDSL**LATEST REVISION**

December 1972

PURPOSE

The package SVDSL determines elements of the singular value decomposition of a given matrix for use in the analysis of the least squares solution of an overdetermined linear system.

That is, given an $M \times N$ matrix T and an n -vector b , an n -vector x (which has minimal length over all n -vectors) is sought, such that

$$\|T*x-b\|_2$$

is minimized.

There are two subroutines in SVDSL. They are SUPRSV and SUPRSL. SUPRSV calculates the singular values, right and left singular vectors of T , and applies the left singular vectors to b . SUPRSL calculates the solution vector x .

ACCESS CARDS

*FORTRAN,S=ULIB,N=SVDSL
*COSY

3.SVDSL.V.2

SPACE REQUIRED	2746 ₈ = (1510 ₁₀) locations
ENTRY POINTS	SUPRSV, SUPRSL
SPECIAL CONDITIONS	The input matrix T for SVDSL.V has to be entered one row at a time. With this requirement, only one row of T needs to be stored. The entire matrix T need never be stored. This allows the decomposition of large overdetermined matrices, which otherwise might not have fit in core.
COMMON BLOCKS	None
I/O	All messages are written using ULIBER. In SUPRSV, if $NN < N*(N+3)$, the subroutine prints the message ***INSUFFICIENT SCRATCH STORAGE HAS BEEN PROVIDED TO SUPRSV.
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Jo Walsh, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN

HISTORY

The two routines, SUPRSV and SUPRSL were written by A. K. Cline, NCAR, in December 1972. They were standardized for the NSSL Library by Jo Walsh, NCAR, in June 1974.

ACCURACY

The routine SUPRSL checks for zero singular values; that is, if S_j is the j^{th} singular value of the input matrix, T , and $S_j^2 \leq \text{EPS}^2$ (where EPS is described in input for subroutine SUPRSL), then S_j is declared to be zero.

In this way, the conditioning of the input matrix is monitored.

PORTABILITY

The routine SUPRSV contains two machine dependent constants.

1. MACHEP is the machine single precision floating-point unit round-off error. On the CDC 6000-7000 series, $\text{MACHEP} = 2^{-47}$.
2. TOL is the ratio of the smallest positive single-precision floating-point number to MACHEP. On the CDC 6000-7000 series, $\text{TOL} = 2^{-929}$.

REQUIRED RESIDENT ROUTINES

ULIBER, RBAIEX, SQRTF

SUBROUTINE SUPRSV (I,ROWI,BI,A,NN,M,N)**DIMENSION OF ARGUMENTS**

ROWI(N),A(NN)

LATEST REVISION

December 1972

PURPOSE

SUPRSV calculates the singular values, right and left singular vectors of an input matrix which has been entered one row at a time. Finally, SUPRSV applies the left singular vectors of the input matrix to the right hand side of an overdetermined linear system.

USAGE

Subroutine SUPRSV should be called once for each row of the input matrix, thus a typical calling sequence is

```

DO 2 I=1,M
DO 1 J=1,N
  .
  .
  (computation of the (I,J) element of the
  input matrix)
  .
  .
1 ROWI(J)=...
  BI=...(element I of the right hand side vector)
2 CALL SUPRSV (I,ROWI,BI,A,NN,M,N)

```

ARGUMENTS**On Input**

I

An integer indicating the index of the row being entered. (I is one on the first call, increases by 1 on each call, and is M on the final call.)

ROWI

A vector of length at least N which on the Ith call contains the N components of row I of the matrix.

BI

A real variable which on the Ith call contains element I of the right hand side vector.

A

A real vector variable with dimension NN which is used internally for scratch storage. The array A should not be altered during the various calls to SUPRSV.

NN

An integer variable set equal to the length of array A. NN should be at least $N*(N+3)$.

M

An integer variable set equal to the total number of rows of the input matrix to be entered.

N

An integer variable set equal to the length of the matrix rows ($N \leq M$).

On Output

After the return from the M^{th} call to SUPRSV, the array A contains the following:

1. $A(1), \dots, A(N*N)$ contain the right singular vectors of the matrix, (i.e., vector J , where $J=1, \dots, N$, is stored in elements $A(1+(J-1)*N), \dots, A(J*N)$.)
2. $A(1+N*N), \dots, A(N*(N+1))$ contain the right hand side vector transformed by the left singular vectors.
3. $A(1+N*(N+1)), \dots, A(N*(N+2))$ contain the singular values of the matrix stored in decreasing order.
4. $A(1+N*(N+2))$ contains the Euclidean norm of the minimal residual.

The quantities I, ROWI, BI, NN, M, and N are never altered.

COMMON BLOCKS

None

ALGORITHM

The algorithm used in SUPRSV for the singular value decomposition is found in G. Golub and C. Reinsch, "Singular Value Decomposition and Linear Least Squares Solutions," *Numer. Math.* 14 (1970), pp 403-420.

SPACE REQUIRED

$2532_8 = (1370_{10})$ locations

TIMING

SUPRSV uses an iterative QR technique for determining the singular values of the input matrix. On the CDC 7600, SUPRSV takes 12 milliseconds to decompose a 30×10 matrix.

PORTABILITY

SUPRSV has two machine dependent constants. These are discussed in Portability for PACKAGE SVDSL.V.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, RBAIEX, SQRTF

SUBROUTINE SUPRSL (N,A,X,EPS)

DIMENSION OF ARGUMENTS A(NN),X(N)

LATEST REVISION December 1972

PURPOSE SUPRSL calculates the solution vectors X of an overdetermined linear system which has been decomposed in subroutine SUPRSV.

USAGE CALL SUPRSL (N,A,X,EPS)

ARGUMENTS

On Input N
 An integer variable set equal to the number of elements in the solution vector.

A
 A real array with dimension NN whose contents are described in output for subroutine SUPRSV.

EPS
 A real variable which estimates the uncertainty in the right hand side vector supplied to subroutine SUPRSV. EPS is the maximum of the Euclidean norms of the differences B-BP, where B is the right hand side vector supplied to SUPRSV, and BP ranges over all acceptable replacements for B. If the elements of B each have independent uncertainty equal to DELTA, then EPS should be SQRT(M)*DELTA.

On Output

X

A real vector variable with dimension at least N. On output, X contains the minimal least squares solution to the problem given to SUPRSV, but altered to allow all right hand sides within EPS of the given vector B.

A(2+N*(N+2))

Contains the actual Euclidean norm difference in B and the right hand side used.

COMMON BLOCKS

None

ALGORITHM

The algorithm is based on the article,

R. J. Hanson and C. L. Lawson, "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems," *Math. Comp.* 23 (1969), pp 787-812.

SPACE REQUIRED
 $205_8 = (133_{10})$ locations
TIMING

The timing is proportional to the quantity N^2+N . On the CDC 7600 SUPRSL takes less than a millisecond to calculate the solution vector of a 30×10 linear system.

PORTABILITY

SUPRSL is portable

REQUIRED RESIDENT ROUTINES

SQRTF

SUBROUTINE TRDI (N,A,B,C,Y,X,KS,WORK)

DIMENSION OF ARGUMENTS A(N),B(N),C(N),Y(N),X(N),WORK(2*N-2)

LATEST REVISION October 1973

PURPOSE TRDI computes the solution of the tridiagonal linear system,

$$\begin{aligned} B(1)*X(1) + C(1)*X(2) &= Y(1) \\ A(I)*X(I-1) + B(I)*X(I) + C(I)*X(I+1) &= Y(I) \\ &I=2,3,\dots,N-1 \\ A(N)*X(N-1) + B(N)*X(N) &= Y(N) \end{aligned}$$

ACCESS CARDS *FORTRAN,S=ULIB,N=TRDI
*COSY

USAGE CALL TRDI (N,A,B,C,Y,X,KS,WORK)

ARGUMENTS

On Input N
The number of unknowns.

A
The subdiagonal of the matrix is stored in locations A(2) through A(N).

B
The diagonal of the matrix is stored in locations B(1) through B(N).

3. TRDI.2

On Input (continued)

C

The super-diagonal of the matrix is stored in locations C(1) through C(N-1).

Y

The right-hand side of the equations is stored in Y(1) through Y(N).

WORK

An array which must be provided for workspace. WORK must be dimensioned at least $2*(N-1)$.

KS

= 0 if this is the initial call to TRDI with a given coefficient matrix.

= 1 if the coefficient matrix is unchanged from the previous call.

On Output

X

An array which contains the solution to the system of equations.

WORK

Contains intermediate values which must not be destroyed if TRDI is to be called again with KS=1.

ENTRY POINTS

TRDI

SPECIAL CONDITIONS

This subroutine assumes the matrix is diagonally dominant. It does not pivot and performs no tests for a singular matrix.

COMMON BLOCKS

None

I/O

No error messages.

PRECISION

Single

REQUIRED ULIB ROUTINES	None
SPECIALIST	Nancy Werner, NCAR, Boulder, Colorado 80302
LANGUAGE	FORTRAN
HISTORY	Standardized October, 1973
ALGORITHM	The system of equations is solved using column-wise Gaussian elimination.
SPACE REQUIRED	$(144)_8 = (100)_{10}$ locations
ACCURACY	Usually equal to machine accuracy, unless the matrix is ill-conditioned.
TIMING	The running time is proportional to n . The time for a matrix of order 50 on the NCAR CDC 7600 is .2 millisecond when $KS = 0$. If $KS = 1$, the time is .1 millisecond.
PORTABILITY	There are no machine dependent constants.
REQUIRED RESIDENT ROUTINES	None
Purpose	For solving a partial differential equation, a difference scheme is often used which involves a diagonally dominant tridiagonal matrix which can be solved by TRDI.

Example

Consider the problem of solving the heat equation

$$\frac{\partial U}{\partial t} = \alpha \frac{\partial^2 U}{\partial x^2}$$

Define $X_J = a + J \left[\frac{\Delta X}{100} \right]$, $J = 0, 1, \dots, 100$
 where

$$\Delta X = \frac{b-a}{100} .$$

U_0 is given at $X_0 = a$

U_{100} is given at $X_{100} = b$

N is the number of unknowns = 99.

Define $t_n = n\Delta t$, $n = 0, 1, \dots$.

Let U_J^n be an approximation to U at X_J and t_n and assume that U_J is given at $t = 0$ for all J . Using a Crank-Nicholson difference scheme, we obtain

$$U_J^{n+1} - U_J^n = \frac{\alpha \Delta t}{2\Delta X^2} [U_{J+1}^{n+1} - 2U_J^{n+1} + U_{J-1}^{n+1} + U_{J+1}^n - 2U_J^n + U_{J-1}^n]$$

or

$$\frac{\alpha \Delta t}{2\Delta X^2} U_{J-1}^{n+1} - \left(\frac{\alpha \Delta t}{\Delta X^2} + 1 \right) U_J^{n+1} + \frac{\alpha \Delta t}{2\Delta X^2} U_{J+1}^{n+1} = Y_J$$

Hence, the input quantities to TRDI would be

$$N = 99$$

$$A(J) = \frac{\alpha \Delta t}{2\Delta X^2} \quad , \quad J = 2, 3, \dots, 99$$

$$B(J) = - \left(\frac{\alpha \Delta t}{\Delta X^2} + 1 \right) \quad , \quad J = 1, 2, \dots, 99$$

$$C(J) = \frac{\alpha \Delta t}{2\Delta X^2} \quad , \quad J = 1, 2, \dots, 98$$

$$Y(J) = -\frac{\alpha\Delta t}{2\Delta X^2} U_{J-1}^n + \left(\frac{\alpha\Delta t}{2\Delta X^2} - 1\right) U_J^n - \frac{\alpha\Delta t}{2\Delta X^2} U_{J+1}^n \quad ,$$

$$J = 1, 2, \dots, 99 \quad .$$

The subroutine call for $n = 1$, $t = \Delta t$ requires that $KS = 0$, while for subsequent calls we use $KS = 1$.

SUBROUTINE TRDIP (N,A,B,C,Y,X,KS,WORK)

DIMENSION OF ARGUMENTS A(N),B(N),C(N),Y(N),X(N),WORK(3*N)

LATEST REVISION October 1973

PURPOSE TRDIP computes the solution of the tridiagonal periodic linear system,

$$\begin{aligned} & B(1)*X(1) + C(1)*X(2) && + A(1)*X(N) = Y(1) \\ & A(I)*X(I-1) + B(I)*X(I) + C(I)*X(I+1) && = Y(I) \\ & \quad \quad \quad I=2,3,\dots,N-1 \\ & C(N)*X(1) + && A(N)*X(N-1) + B(N)*X(N) = Y(N) \end{aligned}$$

ACCESS CARDS *FORTRAN,S=ULIB,N=TRDIP
*COSY

USAGE CALL TRDIP (N,A,B,C,Y,X,KS,WORK)

ARGUMENTS

On Input N
The number of unknowns.

A
The subdiagonal of the matrix is stored in locations A(2) through A(N). A(1) contains the coefficient for X(N) in the first equation.

On Input
(continued)

B

The diagonal of the matrix is stored in locations B(1) through B(N).

C

The super-diagonal of the matrix is stored in locations C(1) through C(N-1). C(N) contains the coefficient for X(1) in the last equation.

Y

The right-hand side of the equation is stored in locations Y(1) through Y(N).

WORK

An array which must be provided for work space. WORK must be dimensioned at least 3*N.

KS

- = 0 if this is the initial call to TRDIP with a given coefficient matrix.
- = 1 if the coefficient matrix is unchanged from the previous call.

On Output

X

An array which contains the solution to the system of equations.

WORK

Contains intermediate values which must not be destroyed if TRDIP is to be called again with KS=1.

ENTRY POINTS

TRDIP

SPECIAL CONDITIONS

This subroutine assumes the matrix to be diagonally dominant. It does not pivot and performs no tests for a singular matrix.

COMMON BLOCKS

None

I/O

No error messages.

PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Nancy Werner, NCAR, Boulder, Colorado 80302
LANGUAGE	FORTRAN
HISTORY	Standardized October, 1973
ALGORITHM	The system of equations is solved using Gaussian elimination.
SPACE REQUIRED	$(316)_8 = (206)_{10}$ locations
ACCURACY	Usually equal to machine accuracy unless the matrix is ill-conditioned.
TIMING	The running time is proportional to n . The time for a matrix of order 50 on the NCAR CEC 7600 is .35 millisecond when $KS=0$. If $KS=1$, the time is .17 millisecond.
PORTABILITY	There are no machine dependent constants.
REQUIRED RESIDENT ROUTINES	None
Purpose	For solving a partial differential equation with periodic boundary conditions, a difference scheme is often used which involves a diagonally dominant tridiagonal matrix with additional matrix elements in each corner. This matrix equation can be solved by TRDIP.

Example

Consider approximating the solution of the heat equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

on the strip $1 \leq x \leq 5$, $t > 0$, with the boundary conditions

$$U(1,t) = U(5,t) .$$

We define a finite difference grid by

$$X_j = 1 + \frac{4}{100} J \quad , \quad J = 0,1,\dots,100$$

$$t_M = M\Delta t \quad , \quad M = 0,1,\dots$$

Let $v_{J,M}$ be an approximation to $U(X_J, t_M)$. Then from the boundary conditions we get

$$v_{0,M} = v_{100,M} \quad \text{and} \quad v_{1,M} = v_{101,M}$$

and we assume that $v_{J,0}$ is given. Using a Crank-Nicholson scheme to approximate the differential equation we get

$$\begin{aligned} & \frac{\alpha\Delta t}{2\Delta x^2} v_{J-1,M+1} - \left(1 + \frac{\alpha\Delta t}{\Delta x^2}\right) v_{J,M+1} + \frac{\alpha\Delta t}{2\Delta x^2} v_{J+1,M+1} \\ & = -\frac{\alpha\Delta t}{2\Delta x^2} v_{J-1,M} - \left(1 - \frac{\alpha\Delta t}{\Delta x^2}\right) v_{J,M} - \frac{\alpha\Delta t}{2\Delta x^2} v_{J+1,M} \end{aligned}$$

for $J = 1,2,\dots,100$, $M = 0,1,2,\dots$.

To solve this linear system using TRDIP we put

$$N = 100$$

$$A(I) = C(I) = \frac{\alpha \Delta t}{2 \Delta x^2}$$

$$B(I) = 1 + \frac{\alpha \Delta t}{\Delta x^2}$$

$$Y(I) = -\frac{\alpha \Delta t}{2 \Delta x^2} v_{I-1, M} - \left(1 - \frac{\alpha \Delta t}{\Delta x^2}\right) v_{I, M} - \frac{\alpha \Delta t}{2 \Delta x^2} v_{I+1, M}$$

$$\left. \begin{array}{l} A(I) = C(I) = \frac{\alpha \Delta t}{2 \Delta x^2} \\ B(I) = 1 + \frac{\alpha \Delta t}{\Delta x^2} \\ Y(I) = -\frac{\alpha \Delta t}{2 \Delta x^2} v_{I-1, M} - \left(1 - \frac{\alpha \Delta t}{\Delta x^2}\right) v_{I, M} - \frac{\alpha \Delta t}{2 \Delta x^2} v_{I+1, M} \end{array} \right\} I = 1, 2, \dots, 100$$

and call TRDIP. For $M=0$, we set $KS=0$, and then for subsequent M we set $KS=1$.

4

QUADRATURE

ADQUAD

GAUSS

SIMPSN

PACKAGE ADQUAD**LATEST REVISION** April 1974**PURPOSE**

To calculate automatically the integral of $F(X)$ over the finite interval (A,B) with relative error less than $EPSIL$. The package contains two routines.

Subroutine QUAD

This implements the basic algorithm which integrates over the whole interval using a sequence of interleaving 1, 3, 7, 15, 31, 63, 127 and 255-point extended Gauss-type quadrature formulae. Since each successive formula employs all points used by its predecessor, no integrand values are wasted when the order of the integration formula is increased. The process is deemed to converge when the relative difference between values of the integral obtained from two successive formulae is less than $EPSIL$.

Function ADQUAD

This control function uses QUAD to integrate over a finite interval employing successive adaptive subdivision if convergence to the required accuracy is not achieved.

4. ADQUAD.2

ACCESS CARDS

*FORTRAN,S=ULIB,N=ADQUAD
*COSY

These cards access the two routines ADQUAD and QUAD.

SPACE REQUIRED

Total space for both routines is $1653_8 = 939_{10}$

ENTRY POINTS

QUAD, ADQUAD.

SPECIAL CONDITIONS

If the range includes discontinuities or singularities, special treatment may be necessary. Discontinuities may be treated by subdividing the interval at each discontinuity. The routine will cope with a singularity of the type X^{**P} , where $0 > P > -.9$ at either or both ends of the range, but the calculation is time consuming, involving many function evaluations.

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB ROUTINES

None

SPECIALIST

Cicely Ridley, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

SUBROUTINE QUAD (A,B,F,RESULT,EPSIL,NPTS,K,IER)

DIMENSION OF ARGUMENTS RESULT(8)

LATEST REVISION April 1974

USAGE CALL QUAD (A,B,F,RESULT,EPSIL,NPTS,K,IER)

ARGUMENTS

On Input

A

Lower limit of integration.

B

Upper limit of integration.

F

F(X) is user written function to calculate the integrand.
F must be declared EXTERNAL in the calling program.

EPSIL

Required relative error of the integral. This should be less than 1×10^{-3} .

On Output

RESULT

Array in which are returned the results of applying 1, 3, 5, 7, 15, 31, 63, 127, and 255-point formulae successively. The number of formulae actually used will depend upon EPSIL. The dimension of RESULT in the calling program should be at least 8.

On Output
(continued)**K**

RESULT(K) contains the value of the integral to the required accuracy. K is determined from the convergence criterion:

$$\text{ABS}(\text{RESULT}(K) - \text{RESULT}(K-1)) \leq \text{EPSIL} * \text{ABS}(\text{RESULT}(K))$$

NPTS

Number of integrand evaluations performed.

IER

Error parameter

= 0 Required accuracy achieved.

= 1 Required accuracy not achieved after working through all eight formulae.

SPACE REQUIRED

$$1177_8 = 639_{10}$$

TIMING

In all but the most trivial examples, the timing is determined by the time, T, needed for an integrand evaluation by the user provided function F(X). The total number, NPTS, of integrand evaluations depends critically upon the behavior of F(X) within the interval (A,B) and may run into thousands for pathological cases. The total time is approximately NPTS*T.

FUNCTION ADQUAD (A,B,F,EPSIL,NPTS,RELERR,IER)

LATEST REVISION April 1974

USAGE R = ADQUAD (A,B,F,EPSIL,NPTS,RELERR,IER)
Where R is the value of the integral.

ARGUMENTS

On Input

A

Lower limit of integration.

B

Upper limit of integration.

F

F(X) is user written function to calculate the integrand.
F must be declared external in the calling program.

EPSIL

Required relative error of the integral. It is recommended
that EPSIL be less than 1×10^{-3} .

On Output

NPTS

Number of integrand evaluations performed.

RELERR

Crude estimate of relative error obtained by summing the
absolute values of the errors produced by QUAD on each
subinterval and dividing by the calculated value of the
integral.

IER

Error parameter.

- = 0 Required accuracy achieved.
- = 1 Relaxed convergence for at least one subinterval. Accuracy should be checked by examining RELEERR. If a subinterval does not converge with relative error EPSIL, a relaxed convergence criterion is applied. This is

$$\text{ABS}(\text{RESULT}(K) - \text{RESULT}(K-1)) \leq \text{ESTIM} * \text{EPSIL}$$

where ESTIM is the estimate of the integral obtained by applying QUAD to the whole interval. This allows for the situation where nearly all the error may be concentrated in one subinterval as for a singularity.

- = 2 Interval stack overflow. Check accuracy. The interval stack allows for 50 intervals needing further subdivision. If the stack is full when another interval needs to be added, that interval is accepted as it stands, even though convergence has failed. This entails loss of accuracy.

SPACE REQUIRED

$454_8 = 300_{10}$

TIMING

See QUAD.

HISTORY

This package is a standardized version of ACM Algorithm 468. Patterson, T.N.L., 1973: Algorithm for automatic numerical integration over a finite interval. *Communications of the ACM, Vol. 16*, pp. 694-699.

ALGORITHM

The basic algorithm used in subroutine QUAD employs a sequence of extended Gauss-type quadrature formulae developed by Patterson. An n-point formula is extended by adding (n+1) interleaving points. The (n+1) abscissae of the new points and the (n) + (n+1) weights for both old and new points are

ALGORITHM
(continued)

selected to maximize the degree of the integrating polynomial to $(3n+2)$. This degree is one greater than one might superficially expect since the mid point is always an optimum abscissa. Starting with a Gauss one point formula, extensions to 3, 7, 15, 31, 63, 127 and 255-point formulae are made successively. Patterson has been able to compute the necessary weights and abscissae to high accuracy. For greater detail see Patterson, T.N.L., 1973: The optimum addition of points to quadrature formulae. *Math. Comp. Vol. 22*, pp. 847-856.

The control function ADQUAD uses adaptive subdivision of the given interval if the required accuracy is not achieved by applying the subroutine QUAD to the whole interval. At each stage of the process an interval is presented for subdivision. The interval is halved and QUAD applied to each subinterval. Should QUAD fail on the first subinterval, this subinterval is stacked for future subdivision and the second subinterval is immediately examined. Should QUAD fail on the second subinterval, this subinterval is immediately subdivided and the whole process is repeated. Each time a converged result is obtained, it is accumulated as the partial value of the integral. When QUAD converges for both subintervals, the interval last stacked is subdivided and the whole process repeated. The process continues until the stack is empty.

ACCURACY

The relative accuracy is defined by the parameter EPSIL. Maximum possible accuracy would be somewhat less than machine accuracy since weights and abscissae are given in single precision.

PORTABILITY

The data statements in the subroutine QUAD are not standard.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

SUBROUTINE GAUSS (KEY,A,B,N,EPS,W,M,C,X,WRK,IER)**DIMENSION OF
ARGUMENTS**

C(N),X(N),WRK(5*M+3*N+10)

LATEST REVISION

October 1973

PURPOSE

GAUSS calculates Gaussian quadrature abscissas and weights relative to a given weight function on a given finite or infinite interval of integration.

ACCESS CARDS

*FORTRAN,S=ULIB,N=GAUSS
*COSY

USAGE

CALL GAUSS (KEY,A,B,N,EPS,W,M,C,X,WRK,IER)

ARGUMENTS

On Input

KEY

Indicates whether lower and upper limits of integration (A and B) are finite or infinite.

- = 0 if A and B are both finite
- = 1 if A=-infinity and B is finite
- = 2 if A is finite and B=infinity
- = 3 if A=-infinity and B=infinity

A

Lower limit of integration. A may have an arbitrary value if KEY=1 or KEY=3.

B

Upper limit of integration. B may have an arbitrary value if KEY=2 or KEY=3.

N

Number of weights and abscissas desired, which is the order of the resulting Gaussian quadrature formula.

EPS

Relative error, used to control termination of iteration for calculation of abscissas. If D significant figures are desired, set $EPS=5.*10.**(-D)$, allowing for a moderate accumulation of round off error.

W

Name of weight function, supplied by the user as a one argument function subroutine. W must be declared EXTERNAL in the calling routine. W should be positive throughout the interval of integration. If W has any singularities, they should be monotonic and located at the endpoints of the interval of integration. If W has a square root singularity, typified by $1/\text{SQRT}(T)$ at $T=0$, more accurate results may be obtained by setting M, described below, negative.

M

The absolute value of M is the number of points used by GAUSS for the discretization of the weight function, W. Try $M=5*N$ and double M until the desired accuracy is achieved. If the user knows the value of the integral (from A to B) of $P(T)*W(T)$ for some polynomial P of degree K, where $0 < K < 2*N-1$, then a good test for whether M has been chosen large enough is to compare this value with the sum (for $I=1$ to N) of $C(I)*P(X(I))$, where the C and X arrays are the returned weights and abscissas. If M is negative, this indicates that W has a square root type singularity at one or both of the endpoints of the integration interval.

WRK

An array which must be provided for workspace. WRK must be dimensioned at least $5*M+3*N+10$.

On Output

C

Contains the N weights for Gaussian quadrature.

X

Contains the N corresponding abscissas for Gaussian quadrature.

IER

= 0 if no errors occurred
 = 1 if algorithm diverged in attempting to calculate the abscissas. A message is printed in this case.
 Either increase EPS, decrease N, or use the double precision version GAUSD.

ENTRY POINTS

GAUSS,GXQ,FEJER,GCHEBY,KEYTR,GORTH,LOLIM,UPLIM,PP

SPECIAL CONDITIONS

None

COMMON BLOCKS	None
I/O	A message is printed if convergence fails.
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	R. K. Rew, NCAR, Boulder, Colorado 80303.
LANGUAGE	FORTRAN
HISTORY	Adapted, modified, and standardized by R. K. Rew from an ALGOL algorithm, CACM331, by W. Gautschi.
ALGORITHM	A discrete inner product with respect to W is defined using M points. The desired abscissas and weights are approximated by the zeroes and weight factors of the resulting discrete orthogonal polynomials. Roots are determined by Newton iterations and deflation followed by refinement using the undeflated polynomial. A more complete description of this algorithm may be found in "Construction of Gauss-Christoffel Quadrature Formulas," by W. Gautschi, <i>Math. Comput.</i> 22 (1968), pp. 251-270.
SPACE REQUIRED	$2013_8 = 1035_{10}$ locations

ACCURACY

The relative error in the weights and abscissas is within EPS if M is large enough. The weights are calculated from the abscissas and are therefore somewhat less accurate.

TIMING

Depends on W, EPS, N, and M. With $W(X)=A\log(1./X)$, $EPS=5.E-10$, $N=40$, and $M=1200$, it took .72 seconds on the CDC 7600.

PORTABILITY

Fully portable. A fifteen significant digit value of PI is contained in a data statement in subroutines GCHEBY and FEJER which can be modified, if desired.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, COS, SQRT

SIMPSN

FUNCTION SIMPSN (X,Y,NUM,IER)

DIMENSION OF ARGUMENTS X(NUM),Y(NUM)

LATEST REVISION May, 1974

PURPOSE To integrate over a given set of ordinates for equally spaced abscissas using Simpson's method or using entry point SIMPNE to integrate analytically a 3-point Lagrangian interpolation polynomial fitting the ordinates.

ACCESS CARDS *FORTRAN,S=ULIB,N=SIMPSN
*COSY

USAGE For equally spaced abscissas, use
YINT = SIMPSN(X,Y,NUM,IER) .

For unequally spaced abscissas, use
YINT = SIMPNE(X,Y,NUM,IER) .

ARGUMENTS**On Input**

X

For SIMPSN, this is a single variable specifying the increment for the equally spaced abscissas.

For SIMPNE, this is a vector of length NUM specifying the unequally spaced abscissas.

Y

Vector of ordinates.

NUM

Length of vector Y (should be > 2).

On Output

IER

Error flag.

= 32 NUM is < 3, no execution.

= 0 No error.

ENTRY POINTS

SIMPSN, SIMPNE

SPECIAL CONDITIONS

If NUM is an even number, Simpson's rule is applied to interval X_1 through X_{NUM-1} and the result is added to the integral over X_{NUM-1} to X_{NUM} of the three-point Lagrangian interpolating polynomial through X_{NUM-2} , X_{NUM-1} , and X_{NUM} .

COMMON BLOCKS

None

I/O

None

PRECISION Single

REQUIRED ULIB
ROUTINES None

SPECIALIST William B. Frye, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Standardized May 1974.

ALGORITHM Simpson's rule and Lagrangian interpolation.

SPACE REQUIRED 351₈

ACCURACY For equally spaced abscissas the error is proportional to $h^4 f^{(4)}(c)$ where C is some point within the limits of integration. For unequally spaced abscissas the error is proportional to the $f^{(4)}(x)$ times the fourth power of the maximum distance between adjacent abscissas.

TIMING Approximately 3 milliseconds for equal abscissas with 9999 and 10000 ordinates. Approximately 76 milliseconds for unequal abscissas with 9999 and 10000 ordinates. Both on CDC 7600.

PORTABILITY

Except for the additional entry point, there are no restrictions. Otherwise, routine complies with American National Standards Institute FORTRAN.

**REQUIRED RESIDENT
ROUTINES**

None

March 1975

NCAR Software Support Library

Volume 2

Editors: Jeanne C. Adams
Alan K. Cline
Margaret A. Drake
Roland A. Sweet

ATMOSPHERIC TECHNOLOGY DIVISION
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
BOULDER, COLORADO

(5) ORDINARY DIFFERENTIAL EQUATIONS

(6) PARTIAL DIFFERENTIAL EQUATIONS

(7) SPECIAL FUNCTIONS

(8) FAST FOURIER TRANSFORMS

(9) STATISTICS/RANDOM NUMBERS

CONTENTS
VOLUME 11

[CHAPTER 1]	SOLUTION OF NON-LINEAR SYSTEMS DETERMINATION OF ROOTS OF POLYNOMIALS	
	DBNDZRO	1.DBNDZRO.1
	DCPOLY	1.DCPOLY.1
	DRPOLY	1.DRPOLY.1
	RTNI	1.RTNI.1
[CHAPTER 2]	INTERPOLATION, APPROXIMATION AND SMOOTHING	
	BSL1NT	2.BSL1NT.1
	BSL2NT	2.BSL2NT.1
	CUBSPL	2.CUBSPL.1
	CURV	2.CURV.1
	HRM1NT	2.HRM1NT.1
	HRM2NT	2.HRM2NT.1
	KURV	2.KURV.1
	KURVP	2.KURVP.1
	QURV	2.QURV.1
	SPLPAK	2.SPLPAK.1
	SURF	2.SURF.1
	TRIANGLE	2.TRIANGLE.1
[CHAPTER 3]	SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/ EIGENVECTOR ANALYSIS	
	BDSLV	3.BDSLV.1
	BND3	3.BND3.1
	CHLSLV	3.CHLSLV.1
	EIGCFA	3.EIGCFA.1
	EIGHFS	3.EIGHFS.1
	EIGRFA	3.EIGRFA.1
	EIGSFM	3.EIGSFM.1
	EIGSFS	3.EIGSFS.1
	EIGSTM	3.EIGSTM.1
	EIGSTS	3.EIGSTS.1
	HSHSLV	3.HSHSLV.1
	INVMTX	3.INVMTX.1
	LINEQSV	3.LINEQSV.1
	SUPRLS	3.SUPRLS.1
	SVDSLVS	3.SVDSLVS.1
	TRDI	3.TRDI.1
	TRDIP	3.TRDIP.1

[CHAPTER 4]	NUMERICAL INTEGRATION (QUADRATURE)	
	ADQUAD	4.ADQUAD.1
	GAUSS	4.GAUSS.1
	SIMPSN	4.SIMPSN.1
CHAPTER 5	SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS	
	AML	5.AML.1
	DIFSUB	5.DIFSUB.1
	GEAR	5.GEAR.1
	RKL	5.RKL.1
CHAPTER 6	SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS	
	BLKTRI	6.BLKTRI.1
	POIS	6.POIS.1
	PWSCRT	6.PWSCRT.1
	PWSCSP	6.PWSCSP.1
	PWSCYL	6.PWSCYL.1
	PWSSSP	6.PWSSSP.1
CHAPTER 7	EVALUATION OF SPECIAL MATHEMATICAL FUNCTIONS	
	BESLIK	7.BESLIK.1
	BESLJY	7.BESLJY.1
	CXERFC	7.CXERFC.1
	ELIPE	7.ELIPE.1
	ELIPK	7.ELIPK.1
	EXPINT	7.EXPINT.1
	HYPER	7.HYPER.1
CHAPTER 8	FAST FOURIER ANALYSIS	
	FFT	8.FFT.1
	FFTPOW2	8.FFTPOW2.1
CHAPTER 9	STATISTICAL ANALYSIS AND RANDOM NUMBER GENERATION	
	CNFPRB	9.CNFPRB.1
	RGRSN1	9.RGRSN1.1
	RGRSN2	9.RGRSN2.1
	RGRSN3	9.RGRSN3.1
	RGRSN4	9.RGRSN4.1
	RGRSN5	9.RGRSN5.1
	RGRSN6	9.RGRSN6.1
	RLHPTS	9.RLHPTS.1
	RNDEV	9.RNDEV.1
	SPAL	9.SPAL.1
	SPECFT	9.SPECFT.1

[CHAPTER 10]	SPECIAL-PURPOSE INPUT/OUTPUT ROUTINES	
	BRANRD	10.BRANRD.1
	CORFOR	10.CORFOR.1
	READLX	10.READLX.1
	TAPECY	10.RAPECY.1
	UBLOK	10.UBLOK.1
	UZBLOK	10.UZBLOK.1
[CHAPTER 11]	DATA PROCESSING UTILITY ROUTINES	
	BSEARCH	11.BSEARCH.1
	CHCONV	11.CHCONV.1
	CONV360	11.CONV360.1
	DATE	11.DATE.1
	HOURS	11.HOURS.1
	LCKSUM	11.LCKSUM.1
	MOVE	11.MOVE.1
	UCHAR	11.UCHAR.1
[CHAPTER 12]	COMPUTER GRAPHICS	
	INTRODUCTION	
	AUTOGRAPH	12.AUTOGRAPH.1
	CONREC Standard	12.CONREC STANDARD.1
	CONRECQCK	12.CONRECQCK.1
	CONRECSMTH	12.CONRECSMTH.1
	DASHCHAR	12.DASHCHAR.1
	DASHLINE	12.DASHLINE.1
	DASHSMTH	12.DASHSMTH.1
	HAFTON	12.HAFTON.1
	ISOSRF	12.ISOSRF.1
	ISOSRFHR	12.ISOSRFHR.1
	PWRX	12.PWRX.1
	PWRY	12.PWRY.1
	PWRZ	12.PWRZ.1
	SCROLL	12.SCROLL.1
	SRFACE	12.SRFACE.1
	SUPMAP	12.SUPMAP.1
	VELVEC	12.VELVEC.1
	APPENDIX 1: PROCESSING ARRAYS	
	APPENDIX 2: TRANSFORMATIONS	
[CHAPTER 13]	FILE MANIPULATION, TEXT EDITING, PROGRAM PREPROCESSING AND DEBUGGING	
	EDITOR	13.EDITOR.1
	FIDEL	13.FIDEL.1
	FLEX	13.FLEX.1
	FRED	13.FRED.1

DESCRIPTION OF SUBROUTINES**VOLUME II****[CHAPTER 1]****SOLUTION OF NON-LINEAR SYSTEMS
DETERMINATION OF ROOTS OF POLYNOMIALS**

- DBNDZRO** Computes a posteriori error bounds and improvements to roots of polynomials.
- DCPOLY** Finds all the roots of a polynomial with complex coefficients using double precision.
- DRPOLY** Finds all the roots of a polynomial with real coefficients using double precision.
- RTNI** Finds a root of a given non-linear equation.

[CHAPTER 2]**INTERPOLATION, APPROXIMATION AND SMOOTHING**

- BSL1NT** Performs one-dimensional Bessel interpolation of selected order for values and derivatives.
- BSL2NT** Performs two-dimensional Bessel interpolation of selected order for values and derivatives.
- CUBSPL** Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives.
- CURV** Performs one-dimensional interpolation, differentiation, and integration using splines under tension.

[CHAPTER 2]
(continued)

INTERPOLATION, APPROXIMATION AND SMOOTHING (continued)

HRMINT	Performs one-dimensional Hermite interpolation of selected order for values and derivatives.
HRM2NT	Performs two-dimensional Hermite interpolation of selected order for values and derivatives.
KURV	Performs interpolation of a parameterized curve in the plane using splines under tension.
KURVP	Performs interpolation of a parameterized closed curve in the plane using splines under tension.
QURV	Performs interpolation of a parameterized curve in space using splines under tension.
SPLPAK	Performs least squares fitting of multidimensional cubic splines to arbitrarily located data.
SURF	Performs two-dimensional interpolation using a bi-spline under tension.
TRIANGLE	Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane.

[CHAPTER 3]

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS

BDSL	Solves a banded linear system.
BND3	Solves a tridiagonal linear system using Gaussian elimination with partial pivoting.
CHLSLV	Solves a real, symmetric, positive definite linear system by Cholesky decomposition.

[CHAPTER 3]
(continued)

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS
(continued)

EIGCFA	Computes all the eigenvalues and selected eigenvectors of a general complex matrix.
EIGHFS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix.
EIGRFA	Computes all the eigenvalues and selected eigenvectors of a general real matrix.
EIGSFM	Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric matrix.
EIGSFS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix.
EIGSTM	Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix.
EIGSTS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix.
HSHSLV	Solves a real overdetermined linear system using Householder transformations.
INVMTX	Computes the inverse of a general real matrix using Gaussian elimination with full pivoting.
LINEQSV	Solves a real linear system using Gaussian elimination with partial pivoting.

[CHAPTER 3]
(continued)

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS
(continued)

- SUPRLS** Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.
- SVDSLV** Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.
- TRDI** Solves a diagonally dominant tridiagonal linear system using Gaussian elimination without pivoting.
- TRDIP** Solves a diagonally dominant periodic tridiagonal linear system using Gaussian elimination without pivoting.

[CHAPTER 4]

NUMERICAL INTEGRATION (QUADRATURE)

- ADQUAD** Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of functional evaluations required.
- GAUSS** Calculates Gaussian quadrature abscissae and weights relative to a given weight function on a given finite or infinite interval of integration.
- SIMPSN** Integrates a function using Simpson's rule if given at equally spaced abscissae or using interpolatory quadratics if given at unequally spaced abscissae.

[CHAPTER 5]

SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

- AMI** Performs one step of the implicit fourth order Adams-Moulton method. No error control.
- DIFSUB** Performs one variable step using various methods. Good for stiff systems. Good error control
- GEAR** Performs one step using Gear's subroutine, DIFSUB. Built-in error control.
- RKI** Performs one step of the explicit fourth order Runge-Kutta-Gill method. No error control.

[CHAPTER 6]

SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

- BLKTRI** Solves a separable elliptic equation.
- POIS** Solves an elliptic equation with variable coefficients on one derivative.
- PWSCRT** Solves two-dimensional Helmholtz equation in Cartesian coordinates.
- PWSCSP** Solves a two-dimensional Helmholtz equation in spherical coordinates assuming no functional dependence on longitude.
- PWSCYL** Solves two-dimensional Helmholtz equation in cylindrical coordinates.
- PWSSSP** Solves two-dimensional Helmholtz equation on the surface of a sphere.

[CHAPTER 7]

EVALUATION OF SPECIAL MATHEMATICAL FUNCTIONS

- BESLIK** Computes modified Bessel functions of the first and second kind for real argument and real order.
- BESLJY** Computes Bessel functions of the first and second kind for real argument and real order.
- CXERFC** Computes the complex complementary error function of a complex argument.
- ELIPE** Computes complete elliptic integrals of the second kind.
- ELIPK** Computes complete elliptic integrals of the first kind.
- EXPINT** Computes exponential integrals.
- HYPER** Computes hyperbolic functions.

[CHAPTER 8]

FAST FOURIER ANALYSIS

- FFT** Computes fast Fourier transforms for data vectors of arbitrary length.
- FFTPOW2** Computes fast Fourier transforms for data vectors whose length is a power of two.

[CHAPTER 9]

STATISTICAL ANALYSIS AND RANDOM NUMBER GENERATION

- CNFPRB** Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients.
- RGRSN1** Calculates a set of regression coefficients for an unweighted regression model.
- RGRSN2** Weights the model with a diagonal matrix of weights.
- RGRSN3** Weights the model with the covariance matrix of the random variables.
- RGRSN4** Similar to RGRSN3 except that the covariance matrix is banded.
- RGRSN5** Weights the model using standard deviations of repeated observations.
- RGRSN6** Weights the model using an estimated covariance matrix from repeated observations.
- RLHPTS** Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable.
- RNDEV** Generates independent random deviates with mean 0 and variance 1.
- SPAL** Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data.
- SPECFT** Calculates a single power spectrum of an input time series or the cross spectrum of two input time series, including cospectrum, quadrature, coherence, and phase.

CHAPTER 10

SPECIAL-PURPOSE INPUT/OUTPUT ROUTINES

BRANRD	Provides buffered random access I/O.
CORFOR	Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted.
READLX	Provides free format data-directed input and program control facilities.
TAPECY	Copies, combines, and edits tapes.
UBLOK	Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries.
UZBLOK	Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries.

CHAPTER 11

DATA PROCESSING UTILITY ROUTINES

BSEARCH	Performs binary search of a table of floating point numbers.
CHCONV	Converts characters in one character set to another character set by indexing a conversion table.
CONV360	Converts IBM 360 (EBCDIC) cards to CDC (BCD) cards.
DATE	Computes date information from number of hours after December 31, 1920.
HOURS	Computes hours since December 31, 1920, from date information.

CHAPTER 11
(continued)

DATA PROCESSING UTILITY ROUTINES *(continued)*

- LCKSUM** Provides a fast checksum of data.
- MOVE** Provides a rapid in-core transfer of data.
- UCHAR** Provides a rapid character retrieval facility.

CHAPTER 12

COMPUTER GRAPHICS

- AUTOGRAPH** Draws and annotates curves or families of curves.
- CONREC Standard** Contours two-dimensional arrays, labelling the contour lines.
- CONRECQCK** Like CONREC Standard, but faster and smaller because contours are unlabelled.
- CONRECSMTH** Like CONREC Standard, but bigger and slower because contours are smoothed as well as labelled.
- DASHCHAR** Software dashed line package with labelling capability.
- DASHLINE** Like DASHCHAR, but smaller and faster because it has no labelling capacity.
- DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed.
- HAFTON** Halftone (gray scale) pictures from a two-dimensional array.
- ISOSRF** Iso-valued surfaces (with hidden lines removed) from a three-dimensional array.

CHAPTER 12
(continued)

COMPUTER GRAPHICS (continued)

ISOSRFHR	Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array.
PWRX	High quality software characters.
PWRY	Simplest software characters.
PWRZ	Three-space characters for use with ISOSRF or SRFACE.
SCROLL	Movie titling package.
SRFACE	Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array.
SUPMAP	Continental outlines and political boundaries in various projections.
VELVEC	Two-dimensional velocity field displayed by drawing arrows from the data locations.

CHAPTER 13

FILE MANIPULATION, TEXT EDITING, PROGRAM PREPROCESSING AND DEBUGGING

EDITOR	Program to maintain a tape library of source card files in a PLIB-compatible form.
FIDEL	Pre-compiler for the language PDELAN (an extension of FORTRAN to facilitate implementation of finite difference approximations to partial differential equations).
FLEX	File management of COSYed PLIB card files.
FRED	Pre-compiler for FORTRAN programs providing conditional compilation, macros, debugging aids, evaluation of subscript expressions and renumbering of statement labels.

**ALPHABETICAL LISTING
OF SUBROUTINES****A****ADQUAD**

Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of functional evaluations required. (Volume I, Chapter 4)

AMI

Performs one step of the implicit fourth order Adams-Moulton method. No error control. (Volume II, Chapter 5)

AUTOGRAPH

Draws and annotates curves or families of curves. (Volume III, Chapter 12)

B**BDSL**

Solves a banded linear system. (Volume I, Chapter 3)

BESLIK

Computes modified Bessel functions of the first and second kind for real argument and real order. (Volume II, Chapter 7)

BESLJY

Computes Bessel functions of the first and second kind for real argument and real order. (Volume II, Chapter 7)

BLKTRI

Solves a separable elliptic equation. (Volume II, Chapter 6)

BND3

Solves a tridiagonal linear system using Gaussian elimination with partial pivoting. (Volume I, Chapter 3)

BRANRD

Provides buffered random access I/O. (Volume III, Chapter 10)

BSEARCH

Performs binary search of a table of floating point numbers. (Volume III, Chapter 11)

BSLINT

Performs one-dimensional Bessel interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

BSL2NT

Performs two-dimensional Bessel interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

C

- CHCONV** Converts characters in one character set to another character set by indexing a conversion table. (Volume III, Chapter 11)
- CHLSLV** Solves a real, symmetric, positive definite linear system by Cholesky decomposition. (Volume I, Chapter 3)
- CNFPRB** Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients. (Volume II, Chapter 9)
- CONV360** Converts IBM 360 (EBCDIC) cards to CDC (BCD) cards. (Volume III, Chapter 11)
- CONREC Standard** Contours two-dimensional arrays, labelling the contour lines. (Volume III, Chapter 12)
- CONRECQCK** Like CONREC Standard, but faster and smaller because contours are unlabelled. (Volume III, Chapter 12)
- CONRECSMTH** Like CONREC Standard, but bigger and slower because contours are smoothed as well as labelled. (Volume III, Chapter 12)
- CORFOR** Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted. (Volume III, Chapter 10)
- CUBSPL** Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives. (Volume I, Chapter 2)
- CURV** Performs one-dimensional interpolation, differentiation, and integration using splines under tension. (Volume I, Chapter 2)
- CXERFC** Computes the complex complementary error function of a complex argument. (Volume II, Chapter 7)

D

- DASHCHAR** Software dashed line package with labelling capability. (Volume III, Chapter 12)
- DASHLINE** Like DASHCHAR, but smaller and faster because it has no labelling capacity. (Volume III, Chapter 12)
- DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed. (Volume III, Chapter 12)
- DATE** Computes date information from number of hours after December 31, 1920. (Volume III, Chapter 11)

D (*continued*)

- DBNDZRO** Computes a posteriori error bounds and improvements to roots of polynomials. (Volume I, Chapter 1)
- DCPOLY** Finds all the roots of a polynomial with complex coefficients using double precision. (Volume I, Chapter 1)
- DIFSUB** Performs one variable step using various methods. Good for stiff systems. Good error control. (Volume II, Chapter 5)
- DRPOLY** Finds all the roots of a polynomial with real coefficients using double precision. (Volume I, Chapter 1)
- E**
- EDITOR** Program to maintain a tape library of source card files in a PLIB-compatible form. (Volume III, Chapter 13)
- EIGCFA** Computes all the eigenvalues and selected eigenvectors of a general complex matrix. (Volume I, Chapter 3)
- EIGHFS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix. (Volume I, Chapter 3)
- EIGRFA** Computes all the eigenvalues and selected eigenvectors of a general real matrix. (Volume I, Chapter 3)
- EIGSFM** Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric matrix. (Volume I, Chapter 3)
- EIGSFS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix. (Volume I, Chapter 3)
- EIGSTM** Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix. (Volume I, Chapter 3)
- EIGSTS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix. (Volume I, Chapter 3)
- ELIPE** Computes complete elliptic integrals of the second kind. (Volume II, Chapter 7)
- ELIPK** Computes complete elliptic integrals of the first kind. (Volume II, Chapter 7)

E *(continued)*

EXPINT Computes exponential integrals. (Volume II, Chapter 7)

F

FFT Computes fast Fourier transforms for data vectors of arbitrary length. (Volume II, Chapter 8)

FFTPOW2 Computes fast Fourier transforms for data vectors whose length is a power of two. (Volume II, Chapter 8)

FIDEL Pre-compiler for the language PDELAN (an extension of FORTRAN to facilitate implementation of finite difference approximations to partial differential equations). (Volume III, Chapter 13)

FLEX File management of COSYed PLIB card files. (Volume III, Chapter 13)

FRED Pre-compiler for FORTRAN programs providing conditional compilation, macros, debugging aids, evaluation of subscript expressions and renumbering of statement labels. (Volume III, Chapter 13)

G

GAUSS Calculates Gaussian quadrature abscissae and weights relative to a given weight function on a given finite or infinite interval of integration. (Volume I, Chapter 4)

GEAR Performs one step using Gear's subroutine, DIFSUB. Built-in error control. (Volume II, Chapter 5)

H

HAFTON Halftone (gray scale) pictures from a two-dimensional array. (Volume III, Chapter 12)

HOURS Computes hours since December 31, 1920, from date information. (Volume III, Chapter 11)

HRMINT Performs one-dimensional Hermite interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

HRM2NT Performs two-dimensional Hermite interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

HSHSLV Solves a real overdetermined linear system using Householder transformations. (Volume I, Chapter 3)

HYPER Computes hyperbolic functions. (Volume II, Chapter 7)

I

INVMTX Computes the inverse of a general real matrix using Gaussian elimination with full pivoting. (Volume I, Chapter 3)

ISOSRF Iso-valued surfaces (with hidden lines removed) from a three-dimensional array. (Volume III, Chapter 12)

ISOSRFHR Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array. (Volume III, Chapter 12)

K

KURV Performs interpolation of a parameterized curve in the plane using splines under tension. (Volume I, Chapter 2)

KURVP Performs interpolation of a parameterized closed curve in the plane using splines under tension. (Volume I, Chapter 2)

L

LCKSUM Provides a fast checksum of data. (Volume III, Chapter 11)

LINEQSV Solves a real linear system using Gaussian elimination with partial pivoting. (Volume I, Chapter 3)

M

MOVE Provides a rapid in-core transfer of data. (Volume III, Chapter 11)

P

POIS Solves an elliptic equation with variable coefficients on one derivative. (Volume II, Chapter 6)

PWRX High quality software characters. (Volume III, Chapter 12)

PWRY Simplest software characters. (Volume III, Chapter 12)

PWRZ Three-space characters for use with ISOSRF or SRFACE. (Volume III, Chapter 12)

PWSCRT Solves two-dimensional Helmholtz equation in Cartesian coordinates. (Volume II, Chapter 6)

PWSCSP Solves a two-dimensional Helmholtz equation in spherical coordinates assuming no functional dependence on longitude. (Volume II, Chapter 6)

P *(continued)*

- PWSCYL** Solves two-dimensional Helmholtz equation in cylindrical coordinates. (Volume II, Chapter 6)
- PWSSSP** Solves two-dimensional Helmholtz equation on the surface of a sphere. (Volume II, Chapter 6)

Q

- QURV** Performs interpolation of a parameterized curve in space using splines under tension. (Volume I, Chapter 2)

R

- READLX** Provides free format data-directed input and program control facilities. (Volume III, Chapter 10)
- RGRSN1** Calculates a set of regression coefficients for an unweighted regression model. (Volume II, Chapter 9)
- RGRSN2** Weights the model with a diagonal matrix of weights. (Volume II, Chapter 9)
- RGRSN3** Weights the model with the covariance matrix of the random variables. (Volume II, Chapter 9)
- RGRSN4** Similar to RGRSN3 except that the covariance matrix is banded. (Volume II, Chapter 9)
- RGRSN5** Weights the model using standard deviations of repeated observations. (Volume II, Chapter 9)
- RGRSN6** Weights the model using an estimated covariance matrix from repeated observations. (Volume II, Chapter 9)
- RK1** Performs one step of the explicit fourth order Runge-Kutta-Gill method. No error control. (Volume II, Chapter 5)
- RLHPTS** Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable. (Volume III, Chapter 9)
- RNDEV** Generates independent random deviates with mean 0 and variance 1. (Volume III, Chapter 9)
- RTNI** Finds a root of a given non-linear equation. (Volume I, Chapter 1)

S

- SCROLL** Movie titling package. (Volume III, Chapter 12)
- SIMPSN** Integrates a function using Simpson's rule if given at equally spaced abscissae or using interpolatory quadratics if given at unequally spaced abscissae. (Volume I, Chapter 4)
- SPAL** Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data. (Volume II, Chapter 9)
- SPECFT** Calculates a single power spectrum of an input time series or the cross spectrum of two input time series, including cospectrum, quadrature, coherence, and phase. (Volume II, Chapter 9)
- SPLPAK** Performs least squares fitting of multidimensional cubic splines to arbitrarily located data. (Volume I, Chapter 2)
- SRFACE** Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array. (Volume III, Chapter 12)
- SUPMAP** Continental outlines and political boundaries in various projections. (Volume III, Chapter 12)
- SUPRLS** Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core. (Volume I, Chapter 3)
- SURF** Performs two-dimensional interpolation using a bi-spline under tension. (Volume I, Chapter 2)
- SVDSL** Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core. (Volume I, Chapter 3)

T

- TAPECY** Copies, combines and edits tapes. (Volume III, Chapter 10)
- TRDI** Solves a diagonally dominant tridiagonal linear system using Gaussian elimination without pivoting. (Volume I, Chapter 3)
- TRDIP** Solves a diagonally dominant periodic tridiagonal linear system using Gaussian elimination without pivoting. (Volume I, Chapter 3)

T *(continued)***TRIANGLE**

Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane. (Volume I, Chapter 2)

U**UBLOK**

Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries. (Volume III, Chapter 10)

UCHAR

Provides a rapid character retrieval facility. (Volume III, Chapter 11)

UZBLOK

Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries. (Volume III, Chapter 10)

V**VELVEC**

Two-dimensional velocity field displayed by drawing arrows from the data locations. (Volume III, Chapter 12)

5

ORDINARY DIFFERENTIAL EQUATIONS

**AM1
DIFSUB
GEAR
RK1**

SUBROUTINE AM1 (N,H,T,Y,DERIV,WK,ISTART)

DIMENSION OF ARGUMENTS Y(N),WK(5*N)

LATEST REVISION December 1, 1973

PURPOSE Given the system of N first order ordinary differential equations

$$\begin{aligned} \text{DY(I)}/\text{DT} &= \text{F(I,T,Y(1),Y(2),\dots,Y(N))} \\ \text{I} &= 1,2,\dots,N \end{aligned}$$

This routine advances the solution by one step (of length H) using the Adams - Moulton 4th order method.

ACCESS CARDS *FORTRAN,S=ULIB,N=AM1
 *COSY

USAGE Preceding the first call, the user must specify N, H, T, Y as discussed below. Also, set

$$\text{ISTART} = 1$$

USAGE*(continued)*

then

CALL AML (N,H,T,Y,DERIV,WK,ISTART)

Upon return, T will have the value $T_{\text{[initial]}} + H$, Y will have the vector value $Y(T_{\text{[initial]}} + H)$ and ISTART will have been set to 2. This means ISTART must be a variable not a literal in the calling program. AML can then be called repeatedly without the user changing any of the arguments. To reinitialize, set ISTART = 1.

ARGUMENTS**On Input**

N

The number of first order ordinary differential equations to be integrated.

H

The step size.

T

The independent variable. Preceding the first call, the user must set T to its initial value.

Y

An array containing the N dependent variables. Preceding the first call, the user must set Y to its initial value.

DERIV

An external subroutine provided by the user to calculate the derivatives. DERIV must be declared EXTERNAL in the calling program.

WK

Work space of dimension at least 5N.

ISTART

- = 1 preceding initial step.
- = 2 for subsequent steps.

On Output

T

$T = T + H$ the updated value of the independent variable.

Y

The input array is replaced by values of the dependent variables updated to $T + H$.

WK

Contains the back values of the solution and its derivatives and must not be disturbed if the user wishes to advance the solution further without reinitializing.

ISTART

Will have been set to 2.

ENTRY POINTS

AM1, RKL

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB
SUBROUTINES

RKL

ROUTINES TO BE
PROVIDED BY USER

SUBROUTINE DERIV (N,T,Y,DY)

On Input

N number of dependent variables.
T independent variable.
Y array of N dependent variables.

On Output

DY array of N derivatives corresponding to input T
and Y values.

The subroutine should evaluate

$$DY(I) = F(I,T,Y(1),Y(2),\dots,Y(N))$$
$$I = 1,2,\dots,N$$

SPECIALIST

Jack Miller, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Standardized December 1, 1973

ALGORITHM

Runge - Kutta - Gill formulas are used for the first 3 steps,
the Adams - Moulton formulas are used to advance the solution.

SPACE REQUIRED405₈ for AM1 + 204₈ for RK1**ACCURACY**

The per step truncation error E has the bound

$$|E| \leq H^5 M$$

where H is the step size and M is a function of F.

TIMING

Timing is dependent mainly on the time spent in DERIV which is called twice per time step.

PORTABILITY

AM1 is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

**REQUIRED RESIDENT
ROUTINES**

None

DIFSUB

**SUBROUTINE DIFSUB (N,T,Y,SAVE,H,HMIN,HMAX,EPS,MF,YMAX,ERROR,KFLAG,
JSTART,MAXDER,PW,SCR1,SCR2)**

DIMENSION OF ARGUMENTS Y(8,N),SAVE(12*N),YMAX(N),ERROR(N),PW(N*N),SCR1(N),SCR2(N)

LATEST REVISION September 20, 1973

PURPOSE Given the system of N first order ordinary differential equations

$$DY(I)/DT=F(I,T,Y(1),Y(2),\dots,Y(N)),I=1,2,\dots,N$$

This routine advances the solution by one step using predictor-corrector methods suitable for stiff or non-stiff systems. The routine selects the method order and step size to maximize the step subject to the given error parameter.

ACCESS CARDS *FORTRAN,S=ULIB,N=DIFSUB
*COSY

USAGE CALL DIFSUB (N,T,Y,SAVE,H,HMIN,HMAX,EPS,MF,YMAX,ERROR,KFLAG,
JSTART,MAXCER,PW,SCR1,SCR2)

ARGUMENTS

On Input

N

The number of first order ordinary differential equations. N may be decreased on later calls if the number of active equations reduces, but it may not be increased without calling with JSTART=0.

On Input*(continued)***T**

The independent variable.

Y

An 8 by N array containing the dependent variables and their scaled derivatives. $Y(J+1,I)$ contains the J^{th} derivative of $Y(I)$ scaled by $H^{**J}/\text{FACTORIAL}(J)$ where H is the current step size. Only $U(1,I)$ need be provided by the calling program on the first entry. If it is desired to interpolate to non-mesh points, these values may be used. If the current step size is H and the value of $T+E$ is needed, form $S=E/H$ and then compute

$$Y(I)(T+E) = \sum_{J=0}^{NQ} Y(J+1,I) * S^{**J}$$

where NQ = current method order returned in $JSTART$ (see below).

SAVEA work space of dimension at least $12*N$.**H**

The step size to be attempted on the next step. H may be adjusted up or down by the program in order to achieve an economical integration. To save computer time, the user is advised to use a fairly small step for the first call. It will be automatically increased later. The routine returns the step size recommended for the next step. This will normally be used as input for the next call.

HMIN

The minimum step size to be used for the integration. Note that on starting, this must be much smaller than the average H expected since a first order method is used initially.

HMAX

The maximum size to which the step may be increased.

EPS

The error test constant. Single step error estimates divided by $YMAX(I)$ must be less than EPS in the Euclidean norm. The step and/or order is adjusted to achieve this.

MF

The method indicator.

- = 0 an Adams predictor-corrector is used.
- = 1 a multi-step method suitable for stiff systems is used. This will also work for non-stiff systems. The user must provide a subroutine PEDERV to calculate the Jacobian, AJ, where AJ is the N by N matrix given by

$$AJ(I,J) = \text{PARTIAL } DF(I,T,Y(1),Y(2),\dots,Y(N))/DY(J)$$
 (This is described in detail below.)
- = 2 The same as case 1, except that the Jacobian is computed by numerical differencing of the derivatives and PEDERV is not used.

YMAX

An array of dimension at least N. Normally each component should be set to 1. for the first entry. The routine returns in YMAX(I) the greatest value of Y(I) seen so far. This is usually used as input for the next call.

JSTART

Input indicator.

- = -1 repeat the last step with a new H.
- = 0 perform the first step. The first step must be done with this value of JSTART so that the routine can initialize itself.
- = 1 take a new step continuing from the last.

On return JSTART=NQ, the current order of the method. JSTART must be reset before each call to DIFSUB.

MAXDER

The maximum derivative (i.e., order) to be used in the method. This must be less than or equal to 7 or 6 for Adams or stiff methods, respectively.

PW

Work space of dimension at least $NMAX^{**2}$, where NMAX is maximum value of N to be used.

SCR1,SCR2

Work spaces each of dimension at least N.

On Output

T

Updated value of the independent variable.

Y

Updated values of the dependent variables and their scaled derivatives.

H

The step size recommended for the next call.

YMAX

YMAX(I) is set to the greatest value of Y(I) seen so far.

ERROR

Array of dimension at least N used to return the estimated absolute one step error in each component.

KFLAG

Completion code.

= 1 the step was successful.

= -1 the step was taken with H=HMIN but the requested error was not achieved.

= -2 the maximum order (MAXDER) specified was found to be too large.

= -3 corrector convergence could not be achieved for H.GT.HMIN.

= -4 requested error is smaller than can be handled for this problem.

JSTART

On return JSTART=NQ, the current order of the method.

ENTRY POINTS

DIFSUB, SOLVEL, DECOMPL

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O None

PRECISION Single

REQUIRED ULIB
ROUTINES None

ROUTINES TO BE
PROVIDED BY USER

DIFFUN

This subroutine evaluates the derivatives of the dependent variables, $Y(I)$, with respect to the independent variable T .

SUBROUTINE DIFFUN(T,Y,DY)

On Input:

T independent variable
Y 8 by N array. Dependent variables are in $Y(1,I)$,
 $I = 1,2,\dots,N$.

On Output:

DY an array of dimension at least N. On return

$DY(I) = F(I,T,Y(1),Y(2),\dots,Y(N)), I = 1,2,\dots,N$.

Notes:

1. The subroutine DIFFUN should include the dimension statement

DIMENSION Y(8,1),DY(1)

2. The name and argument list of the user provided routine for calculating the derivatives are *not compatible* with other *NCAR Scientific Subroutine Library* routines for solving ordinary differential equations.

ROUTINES TO BE
PROVIDED BY USER
(continued)

PEDERV

When the method indicator, MF=1, a sub
must be provided to calculate the N by

$$AJ(I,J) = \text{PARTIAL DF}(I,T,Y(1),Y(2))$$

$$I,J = 1,2,\dots,N.$$

If MF = 0 or 2, a dummy PEDERV routine
the loader. This may consist of the t
RETURN, END.

CALL PEDERV(T,Y,PW,M)

On Input:

T independent variable.

Y 8 by N array. Dependent variable
Y(1,I), I = 1,2,...,N.

M the first dimension of array PW i
program. M must be greater than

On Output:

PW two-dimensional array in which th
returned. This routine must plac
element AJ(I,J) in PW(I,J) for I,J

Note: The routine should include the

DIMENSION Y(8,1),PW(M,1)

SPECIALIST

Documentation standardized, September 19
NCAR, Boulder, Colorado 80302.

LANGUAGE

FORTRAN

HISTORY

The original version of this routine was
Department of Computer Science, Universi

ALGORITHM

Variable order Adams-Moulton and Gear fo

SPACE REQUIRED

2563₈

SUBROUTINE GEAR (N,H,T,Y,DERIV,WK,ERR,ISTART,ISTIFF,IER)**DIMENSION OF
ARGUMENTS**

Y(N),WK((25+N)*N)

LATEST REVISION

September 14, 1973

PURPOSE

To advance the solution of a first order ordinary differential equation system, $DY/DT=F(T,Y)$, by one tabulation interval using GEAR's routine DIFSUB. Suitable for stiff or well behaved systems.

ACCESS CARDS

*FORTRAN,S=ULIB,N=GEAR
*COSY

USAGE

CALL GEAR (N,H,T,Y,DERIV,WK,ERR,ISTART,ISTIFF,IER)

ARGUMENTS**On Input**

N

Number of first order equations.

H

Desired tabulation interval.

T

Independent variable.

On Input
(continued)

Y

Array of N dependent variables. Dimension of Y is calling program must be greater than or equal to N.

DERIV

External subroutine provided by the user to calculate derivative array, F. Must be declared EXTERNAL in calling program. (See below for detailed description.)

ERR

Error control parameter. If the function is increasing, the Euclidean norm of the relative one-step errors of the dependent variables is to be less than ERR. If the function is decreasing, absolute errors are used. The user wishing to employ absolute errors for an increasing function, or relative errors for a decreasing one, should use DIFSUB.

ISTIFF

= 1 equations are well behaved.

= 2 equations are stiff.

ISTART

= 1 first tabulation interval.

= 2 second and subsequent tabulation intervals.

WK

Work space for dimension at least $(25+N)*N$.

On Output

T

$T=T+H$, the updated value of the independent variable.

Y

Input Y array is replaced by values updated to $(T+H)$.

IER

Error flag.

- = 0 integration was successful.
- = 33 required accuracy not achieved with integration step greater than or equal to $H*1.E-10$.
- = 34 corrector convergence not achieved with integration step greater than or equal to $H*1.E-10$.
- = 35 requested error is too small for given machine precision.
- = 36 number of function evaluations exceeds allowed maximum (5000).

ENTRY POINTS

GEAR, PEDERV, DIFSUB, DECOMPL, SOLVEL

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

ROUTINES TO BE
PROVIDED BY USER

DERIV

SUBROUTINE DERIV (N,T,Y,DY)

On Input:

N number of dependent variables.

T independent variable.

Y array of N dependent variables.

On Output:

DY array of N derivatives corresponding to input T and Y values.

The subroutine should evaluate

$$DY(I) = F(I,T,Y(1),Y(2),\dots,Y(N)) \text{ for } I = 1,2,\dots,N.$$

SPECIALIST

E. Cicely Ridley, NCAR, Boulder, Colorado 80302

LANGUAGE

FORTRAN

HISTORY

Written and standardized September 1973.

ALGORITHM

This routine uses GEAR's DIFSUB assuming reasonable values for some parameters and interpolating to desired tabular points.

SPACE REQUIRED

The complete ULIB file GEAR requires 3407₈.

ACCURACY

The subroutine gives an error return if the accuracy specified by the parameter ERR is not achieved.

TIMING

This is very variable depending not only upon the number and complexity of the equations, but also upon the behavior of the solution and the chosen values of H and ERR.

PORTABILITY The routine GEAR is portable but DIFSUB, included in the file, has a block of clearly commented machine dependent constants.

**REQUIRED RESIDENT
ROUTINES** None

Purpose GEAR's routine, DIFSUB, uses predictor corrector methods to solve a system of first order ordinary differential equations. For well behaved systems, Adams-Moulton formulae up to the seventh order are used. For stiff systems, special formulae developed by GEAR are employed. Here, the maximum order available is 6. The routine selects the method-order and integration step in such a way as to maximize the step size subject to the given error parameter. The routine has a long parameter list and is somewhat cumbersome to use, at least initially. It also produces results at values of the independent-variable selected by itself, rather than at tabular intervals desired by the user.

Subroutine GEAR is intended to provide an easy-to-use driver for the casual user of DIFSUB. By assuming reasonable values for some parameters, the parameter list is reduced and made more consistent with other ULIB ordinary differential equation routines. Interpolation to desired tabular points is performed automatically.

Algorithm For a detailed description of the algorithm employed by DIFSUB, the user is referred to the DIFSUB writeup. Since DIFSUB stores information from each step in the form of the function value and its scaled derivatives up to order p , where p is the order of the method, the subroutine GEAR carries out interpolation using a Taylor series truncated after the term of order p .

Special Notes

- The version of DIFSUB used by the routine GEAR is slightly different from the routine on ULIB. It is included in the file, GEAR.
- Since the stiff methods will work on well behaved systems, use ISTIFF=2 when in doubt.
- Error flag values and suggested remedies:

IER = 33 or 34, reduce H.

IER = 35, increase ERR.

IER = 36, a limit of 5000 function evaluations per tabulation interval prevents excessive iteration. If ISTIFF=1, change to ISTIFF=2. If ISTIFF=2 already, reduce H.

Example

This example is taken from GEAR's paper on his routine, DIFSUB. (See DIFSUB write-up.)

Consider the stiff system

$$\frac{d\vec{y}}{dt} = U\vec{z} - UBU\vec{y}$$

where

$$U = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix},$$

$$B = \begin{bmatrix} \beta_1 & 0 & 0 & 0 \\ 0 & \beta_2 & 0 & 0 \\ 0 & 0 & \beta_3 & 0 \\ 0 & 0 & 0 & \beta_4 \end{bmatrix},$$

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \quad \vec{z} = \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \quad \vec{w} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = U\vec{y}.$$

The system is to be integrated from $t = 0.0$ to $t = 6.0$ with tabular intervals of $H = 0.1$. Initial conditions are

$$\vec{y}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

and B is defined by $\beta_1 = 1000$, $\beta_2 = 800$, $\beta_3 = -10$, $\beta_4 = .001$.

The analytic solution of the problem is

$$\vec{y} = U \begin{bmatrix} \beta_1 / (1 - (1 + \beta_1)e^{\beta_1 t}) \\ \beta_2 / (1 - (1 + \beta_2)e^{\beta_2 t}) \\ \beta_3 / (1 - (1 + \beta_3)e^{\beta_3 t}) \\ \beta_4 / (1 - (1 + \beta_4)e^{\beta_4 t}) \end{bmatrix}$$

The problem is coded below. After each call to GEAR the correct solution and maximum component errors are computed. Y is the solution produced by the routine GEAR, YC is the correct solution. These results are printed.

5.GEAR.8

```

*FORTRAN,FL
CARD      APPROXIMATE
NUMBER    PROGRAM LOCATION
1          0          PROGRAM TGEAR
2          0          C EXAMPLE OF USE OF INTEGRATION ROUTINE GEAR.
3          0          DIMENSION Y(4),B(4),U(4,4),WK(120),YC(4),Z(4)
4          0          DATA
5          0          1U/-0.5,.5,.5,.5,.5,-0.5,.5,.5,.5,-0.5,.5,.5,.5,-0.5/,
6          0          2B/1.E+3,8.E+2,-10.,1.E-3/
7          0          C ROUTINE WHICH CALCULATES DERIVATIVE ARRAY MUST BE DECLARED EXTERNAL.
8          0          EXTERNAL DERIV
9          0          C INITIALIZATION
10         0          H=.1
11         230         ERR=1.E-3
12         230         T=0.
13         230         DO 1 I=1,4
14         230         Y(I)=-1.
15         230         YC(I)=-1.
16         230         1 CONTINUE
17         241         ISTART=1
18         241         ISTIFF=2
19         241         ERROR=0.
20         241         WRITE(6,102)
21         250         102 FORMAT(1H1,11X,*T*,8X,*Y(1)*,7X,*YC(1)*,8X,*Y(2)*,7X,*YC(2)*,8X,
22         250         1*Y(3)*,7X,*YC(3)*,8X,*Y(4)*,7X,*YC(4)*,7X,*ERROR*)
23         250         WRITE(6,100)T,(Y(I),YC(I),I=1,4),ERROR
24         250         C INTEGRATE OVER 60 TABULATION INTERVALS.
25         271         DO 2 N=1,60
26         271         CALL GEAR(4,H,T,Y,DERIV,WK,ERR,ISTART,ISTIFF,IER)
27         306         ISTART=2
28         306         C CHECK FOR SUCCESSFUL INTEGRATION STEP.
29         311         IF(IER,GT,32) GO TO 3
30         311         C COMPUTE CORRECT RESULT AND ERROR
31         313         DO 4 I=1,4
32         313         GO TO (5,5,6,5), I
33         313         5 CONTINUE
34         313         EXPB=EXP(-B(I)*T)
35         313         Z(I)=B(I)*EXPB/(EXPB-1.-B(I))
36         313         GO TO 4
37         313         6 CONTINUE
38         313         EXPB=EXP(B(I)*T)
39         313         Z(I)=B(I)/(1.-(1.+B(I))*EXPB)
40         313         4 CONTINUE
41         347         ERROR=0.
42         347         DO 7 I=1,4
43         347         YC(I)=0.
44         347         DO 8 J=1,4
45         351         YC(I)=YC(I)+U(I,J)*Z(J)
46         351         8 CONTINUE
47         365         ERROR=AMAX1(ERROR,ABS(Y(I)-YC(I)))
48         365         7 CONTINUE
49         376         WRITE(6,100) T,(Y(I),YC(I),I=1,4),ERROR
50         420         100 FORMAT(1H 10E12.4)
51         420         2 CONTINUE
52         422         STOP
53         424         3 CONTINUE
54         424         WRITE(6,101)
55         427         101 FORMAT(* INTEGRATION ABANDONED*)
56         427         STOP
57         430         END

```

CARD NUMBER	APPROXIMATE PROGRAM LOCATION	CODE
1	0	SUBROUTINE DERIV(N,T,Y,OY)
2	0	C THIS SUBROUTINE CALCULATES THE DERIVATIVE ARRAY.
3	0	DIMENSION U(4,4),Y(4),OY(4),W(4),Z(4),B(4)
4	0	DATA
5	0	1U/-.5,.5,.5,.5,.5,-.5,.5,.5,.5,-.5,.5,.5,.5,-.5/
6	0	2B/1.E+3,8.E+2,-10.,1.E-3/
7	0	DO 1 I=1,N
8	36	W(I)=0.
9	36	DO 2 J=1,N
10	37	W(I)=W(I)+U(I,J)*Y(J)
11	37	2 CONTINUE
12	52	Z(I)=W(I)**2
13	52	1 CONTINUE
14	60	DO 3 I=1,N
15	60	DO 3 J=1,N
16	62	Z(I)=Z(I)-B(I)*U(I,J)*Y(J)
17	62	3 CONTINUE
18	100	DO 4 I=1,N
19	100	OY(I)=0.
20	100	DO 4 J=1,N
21	101	OY(I)=OY(I)+U(I,J)*Z(J)
22	101	4 CONTINUE
23	120	RETURN
24	124	END

T	Y(1)	YC(1)	Y(2)	YC(2)	Y(3)	YC(3)	Y(4)	YC(4)	ERROR
0.0	-1.0000E+00	0.0							
1.0000E-01	-1.6149E+00	-1.6143E+00	-1.6149E+00	-1.6143E+00	7.0591E-01	7.0534E-01	-7.0591E-01	-7.0534E-01	5.7027E-04
2.0000E-01	-2.6681E+00	-2.6709E+00	-2.6681E+00	-2.6709E+00	1.8350E+00	1.8377E+00	-1.8350E+00	-1.8377E+00	2.7335E-03
3.0000E-01	-3.8346E+00	-3.8374E+00	-3.8346E+00	-3.8374E+00	3.0656E+00	3.0683E+00	-3.0656E+00	-3.0683E+00	2.7517E-03
4.0000E-01	-4.6489E+00	-4.6495E+00	-4.6489E+00	-4.6495E+00	3.9349E+00	3.9354E+00	-3.9349E+00	-3.9354E+00	5.2354E-04
5.0000E-01	-5.0455E+00	-5.0473E+00	-5.0455E+00	-5.0473E+00	4.3791E+00	4.3809E+00	-4.3791E+00	-4.3809E+00	1.8342E-03
6.0000E-01	-5.2019E+00	-5.2032E+00	-5.2019E+00	-5.2032E+00	4.5772E+00	4.5785E+00	-4.5772E+00	-4.5785E+00	1.3507E-03
7.0000E-01	-5.2526E+00	-5.2533E+00	-5.2526E+00	-5.2533E+00	4.6647E+00	4.6653E+00	-4.6647E+00	-4.6653E+00	6.6324E-04
8.0000E-01	-5.2628E+00	-5.2626E+00	-5.2628E+00	-5.2626E+00	4.7076E+00	4.7073E+00	-4.7076E+00	-4.7073E+00	2.2124E-04
9.0000E-01	-5.2568E+00	-5.2574E+00	-5.2568E+00	-5.2574E+00	4.7309E+00	4.7315E+00	-4.7309E+00	-4.7315E+00	6.2686E-04
1.0000E+00	-5.2484E+00	-5.2478E+00	-5.2484E+00	-5.2478E+00	4.7488E+00	4.7481E+00	-4.7488E+00	-4.7481E+00	5.2354E-04
1.1000E+00	-5.2390E+00	-5.2372E+00	-5.2390E+00	-5.2372E+00	4.7632E+00	4.7613E+00	-4.7632E+00	-4.7613E+00	1.8557E-03
1.2000E+00	-5.2290E+00	-5.2290E+00	-5.2290E+00	-5.2290E+00	4.7749E+00	4.7726E+00	-4.7749E+00	-4.7726E+00	2.2434E-03
1.3000E+00	-5.2177E+00	-5.2171E+00	-5.2177E+00	-5.2171E+00	4.7834E+00	4.7827E+00	-4.7834E+00	-4.7827E+00	6.4621E-04
1.4000E+00	-5.2080E+00	-5.2081E+00	-5.2080E+00	-5.2081E+00	4.7917E+00	4.7918E+00	-4.7917E+00	-4.7918E+00	1.2252E-04
1.5000E+00	-5.1997E+00	-5.1998E+00	-5.1997E+00	-5.1998E+00	4.8000E+00	4.8002E+00	-4.8000E+00	-4.8002E+00	1.6330E-04
1.6000E+00	-5.1920E+00	-5.1921E+00	-5.1920E+00	-5.1921E+00	4.8078E+00	4.8079E+00	-4.8078E+00	-4.8079E+00	1.3605E-04
1.7000E+00	-5.1848E+00	-5.1850E+00	-5.1848E+00	-5.1850E+00	4.8146E+00	4.8150E+00	-4.8146E+00	-4.8150E+00	3.8249E-04
1.8000E+00	-5.1782E+00	-5.1782E+00	-5.1782E+00	-5.1782E+00	4.8217E+00	4.8217E+00	-4.8217E+00	-4.8217E+00	1.1826E-04
1.9000E+00	-5.1720E+00	-5.1722E+00	-5.1720E+00	-5.1722E+00	4.8274E+00	4.8278E+00	-4.8274E+00	-4.8278E+00	3.7430E-04
2.0000E+00	-5.1661E+00	-5.1664E+00	-5.1661E+00	-5.1664E+00	4.8340E+00	4.8336E+00	-4.8340E+00	-4.8336E+00	4.0530E-04
2.1000E+00	-5.1608E+00	-5.1611E+00	-5.1608E+00	-5.1611E+00	4.8391E+00	4.8389E+00	-4.8391E+00	-4.8389E+00	2.8115E-04
2.2000E+00	-5.1559E+00	-5.1560E+00	-5.1559E+00	-5.1560E+00	4.8438E+00	4.8440E+00	-4.8438E+00	-4.8440E+00	1.1676E-04
2.3000E+00	-5.1508E+00	-5.1513E+00	-5.1508E+00	-5.1513E+00	4.8492E+00	4.8487E+00	-4.8492E+00	-4.8487E+00	4.9404E-04
2.4000E+00	-5.1465E+00	-5.1468E+00	-5.1465E+00	-5.1468E+00	4.8535E+00	4.8532E+00	-4.8535E+00	-4.8532E+00	3.5880E-04
2.5000E+00	-5.1425E+00	-5.1426E+00	-5.1425E+00	-5.1426E+00	4.8574E+00	4.8574E+00	-4.8574E+00	-4.8574E+00	1.1676E-04
2.6000E+00	-5.1389E+00	-5.1387E+00	-5.1389E+00	-5.1387E+00	4.8609E+00	4.8613E+00	-4.8609E+00	-4.8613E+00	4.3716E-04
2.7000E+00	-5.1357E+00	-5.1349E+00	-5.1357E+00	-5.1349E+00	4.8640E+00	4.8631E+00	-4.8640E+00	-4.8631E+00	1.0747E-03
2.8000E+00	-5.1305E+00	-5.1313E+00	-5.1305E+00	-5.1313E+00	4.8696E+00	4.8687E+00	-4.8696E+00	-4.8687E+00	9.2871E-04
2.9000E+00	-5.1273E+00	-5.1280E+00	-5.1273E+00	-5.1280E+00	4.8728E+00	4.8720E+00	-4.8728E+00	-4.8720E+00	7.3533E-04
3.0000E+00	-5.1244E+00	-5.1248E+00	-5.1244E+00	-5.1248E+00	4.8757E+00	4.8752E+00	-4.8757E+00	-4.8752E+00	4.2138E-04
3.1000E+00	-5.1218E+00	-5.1217E+00	-5.1218E+00	-5.1217E+00	4.8783E+00	4.8783E+00	-4.8783E+00	-4.8783E+00	1.1754E-04
3.2000E+00	-5.1176E+00	-5.1188E+00	-5.1176E+00	-5.1188E+00	4.8825E+00	4.8812E+00	-4.8825E+00	-4.8812E+00	1.2786E-03
3.3000E+00	-5.1149E+00	-5.1160E+00	-5.1149E+00	-5.1160E+00	4.8851E+00	4.8840E+00	-4.8851E+00	-4.8840E+00	1.1627E-03
3.4000E+00	-5.1125E+00	-5.1134E+00	-5.1125E+00	-5.1134E+00	4.8876E+00	4.8866E+00	-4.8876E+00	-4.8866E+00	9.6522E-04
3.5000E+00	-5.1102E+00	-5.1109E+00	-5.1102E+00	-5.1109E+00	4.8898E+00	4.8891E+00	-4.8898E+00	-4.8891E+00	6.7776E-04
3.6000E+00	-5.1082E+00	-5.1085E+00	-5.1082E+00	-5.1085E+00	4.8918E+00	4.8915E+00	-4.8918E+00	-4.8915E+00	2.9268E-04
3.7000E+00	-5.1048E+00	-5.1061E+00	-5.1048E+00	-5.1061E+00	4.8952E+00	4.8939E+00	-4.8952E+00	-4.8939E+00	1.3505E-03
3.8000E+00	-5.1027E+00	-5.1039E+00	-5.1027E+00	-5.1039E+00	4.8973E+00	4.8961E+00	-4.8973E+00	-4.8961E+00	1.2397E-03
3.9000E+00	-5.1007E+00	-5.1018E+00	-5.1007E+00	-5.1018E+00	4.8993E+00	4.8982E+00	-4.8993E+00	-4.8982E+00	1.0925E-03
4.0000E+00	-5.0988E+00	-5.0998E+00	-5.0988E+00	-5.0998E+00	4.9012E+00	4.9002E+00	-4.9012E+00	-4.9002E+00	9.2423E-04
4.1000E+00	-5.0971E+00	-5.0978E+00	-5.0971E+00	-5.0978E+00	4.9029E+00	4.9022E+00	-4.9029E+00	-4.9022E+00	7.3750E-04
4.2000E+00	-5.0954E+00	-5.0959E+00	-5.0954E+00	-5.0959E+00	4.9046E+00	4.9041E+00	-4.9046E+00	-4.9041E+00	5.4019E-04
4.3000E+00	-5.0938E+00	-5.0941E+00	-5.0938E+00	-5.0941E+00	4.9062E+00	4.9059E+00	-4.9062E+00	-4.9059E+00	3.4122E-04
4.4000E+00	-5.0915E+00	-5.0924E+00	-5.0915E+00	-5.0924E+00	4.9085E+00	4.9076E+00	-4.9085E+00	-4.9076E+00	8.7229E-04
4.5000E+00	-5.0898E+00	-5.0907E+00	-5.0898E+00	-5.0907E+00	4.9102E+00	4.9093E+00	-4.9102E+00	-4.9093E+00	8.8771E-04
4.6000E+00	-5.0881E+00	-5.0890E+00	-5.0881E+00	-5.0890E+00	4.9119E+00	4.9110E+00	-4.9119E+00	-4.9110E+00	9.5904E-04
4.7000E+00	-5.0864E+00	-5.0875E+00	-5.0864E+00	-5.0875E+00	4.9136E+00	4.9125E+00	-4.9136E+00	-4.9125E+00	1.0971E-03
4.8000E+00	-5.0847E+00	-5.0860E+00	-5.0847E+00	-5.0860E+00	4.9153E+00	4.9140E+00	-4.9153E+00	-4.9140E+00	1.3140E-03
4.9000E+00	-5.0829E+00	-5.0845E+00	-5.0829E+00	-5.0845E+00	4.9155E+00	4.9155E+00	-4.9155E+00	-4.9155E+00	1.6219E-03
5.0000E+00	-5.0811E+00	-5.0831E+00	-5.0811E+00	-5.0831E+00	4.9189E+00	4.9169E+00	-4.9189E+00	-4.9169E+00	2.0333E-03
5.1000E+00	-5.0815E+00	-5.0817E+00	-5.0815E+00	-5.0817E+00	4.9185E+00	4.9183E+00	-4.9185E+00	-4.9183E+00	2.3501E-04
5.2000E+00	-5.0801E+00	-5.0804E+00	-5.0801E+00	-5.0804E+00	4.9196E+00	4.9196E+00	-4.9196E+00	-4.9196E+00	2.7596E-04
5.3000E+00	-5.0788E+00	-5.0791E+00	-5.0788E+00	-5.0791E+00	4.9212E+00	4.9209E+00	-4.9212E+00	-4.9209E+00	3.6810E-04
5.4000E+00	-5.0774E+00	-5.0779E+00	-5.0774E+00	-5.0779E+00	4.9226E+00	4.9221E+00	-4.9226E+00	-4.9221E+00	5.1866E-04
5.5000E+00	-5.0759E+00	-5.0767E+00	-5.0759E+00	-5.0767E+00	4.9241E+00	4.9233E+00	-4.9241E+00	-4.9233E+00	7.3496E-04
5.6000E+00	-5.0745E+00	-5.0755E+00	-5.0745E+00	-5.0755E+00	4.9255E+00	4.9245E+00	-4.9255E+00	-4.9245E+00	1.0244E-03
5.7000E+00	-5.0730E+00	-5.0744E+00	-5.0730E+00	-5.0744E+00	4.9270E+00	4.9256E+00	-4.9270E+00	-4.9256E+00	1.3946E-03
5.8000E+00	-5.0734E+00	-5.0733E+00	-5.0734E+00	-5.0733E+00	4.9266E+00	4.9267E+00	-4.9266E+00	-4.9267E+00	1.2684E-04
5.9000E+00	-5.0723E+00	-5.0722E+00	-5.0723E+00	-5.0722E+00	4.9277E+00	4.9278E+00	-4.9277E+00	-4.9278E+00	9.6659E-05
6.0000E+00	-5.0712E+00	-5.0712E+00	-5.0712E+00	-5.0712E+00	4.9288E+00	4.9288E+00	-4.9288E+00	-4.9288E+00	3.9861E-05

STOP

SUBROUTINE RK1 (N,H,T,Y,DERIV,WK,ISTART)

DIMENSION OF ARGUMENTS Y(N),WK(2*N)

LATEST REVISION December 1, 1973

PURPOSE Given the system of N first order ordinary differential equations

$$\begin{aligned} \text{DY(I)/DT} &= \text{F(I,T,Y(1),Y(2),\dots,Y(N))}, \\ \text{I} &= 1,2,\dots,N \end{aligned}$$

This routine advances the solution by one step (of length H) using Gill's modification of the 4th order Runge - Kutta method.

ACCESS CARDS *FORTRAN,S=ULIB,N=RK1
 *COSY

USAGE Preceding the first call, the user must specify N, H, T, Y as discussed below. Also set

 ISTART = 1

USAGE*(continued)*

then

CALL RK1 (N,H,T,Y,DERIV,WK,ISTART)

Upon return T will have the value $T_{[initial]} + H$, Y will have the vector value $Y(T_{[initial]} + H)$ and ISTART will have the value 2. (This means ISTART must be a variable not a literal.) RK1 can then be called repeatedly without the user changing any of the arguments. To reinitialize, set ISTART = 1.

ARGUMENTS**On Input**

N

The number of first order ordinary differential equations to be integrated.

H

The step size.

T

The independent variable. Preceding first call, the user must set T to its initial value.

Y

An array containing the N dependent variables. Preceding the first call, the user must set Y to its initial value.

DERIV

An external subroutine provided by the user to calculate the derivatives. DERIV must be declared EXTERNAL in the calling subroutine.

WK

Work space of dimension at least 2N.

ISTART

= 1 for initial step.

= 2 for subsequent steps.

On Output

T

T = T+H, the updated value of the independent variable.

Y

Input array Y is replaced by values of dependent variables updated to T+H.

WK

$WK(I,1) = dY(I)/dt \quad I = 1,2,\dots,N$

WK(I,2) contains Gill's Q's (see reference).

WK must not be destroyed if the user wishes to advance the solution further without reinitializing.

ISTART

Will have been set to 2.

ENTRY POINTS

RK1

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

ROUTINES TO BE
PROVIDED BY USER

SUBROUTINE DERIV (N,T,Y,DY)

On Input

N number of dependent variables.

T independent variable.

Y array of N dependent variables.

On Output

DY array of N derivatives corresponding to input
and Y values.

The subroutine should evaluate

$$DY(I) = F(I,T,Y(1),\dots,Y(N)) \quad I = 1,\dots,N \quad .$$

SPECIALIST

Jack Miller, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

In [1] Gill proposed a Runge-Kutta formula which very cleverly manages to save information about how much roundoff error is made at certain steps in the computation and makes use of this information to reduce the overall roundoff-error. The algorithm Gill developed was for use with fixed point arithmetic. In [2] Thompson gives an adaptation of the Runge-Kutta-Gill scheme for floating point operations. The Thompson adaptation is used here.

1. Gill, S.: *A Process for the Step-by-Step Integration of Differential Equations in an Automatic Computing Machine*. Proc. Cambridge Philos. Soc. 47 (1951), 96-108.
2. Thompson, Robert J.: *Improving Roundoff in Runge-Kutta Computations with Gill's Method*. Comm. ACM 13 (Dec. 1970), 739-740.

ALGORITHM

Fourth order Runge-Kutta-Gill Formulas are used.

SPACE REQUIRED

204₈

ACCURACY

The per step truncation error E has the bound

$$|E| \leq H^5 M$$

where H is the step size and M is a function of f. In cases where $y + E = y$ to machine accuracy, the Runge-Kutta-Gill algorithm allows the roundoff error to grow only very slowly.

TIMING

Timing is dependent mainly on the time spent in DERIV

5.RKL.6

PORTABILITY RKL is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

REQUIRED RESIDENT ROUTINES SQRT

PARTIAL DIFFERENTIAL EQUATIONS

BLKTRI

POIS

PWSCRT

PWSCSP

PWSCYL

PWSSSP

BLKTTRI**SUBROUTINE BLKTTRI (IFLG, NP, N, AN, BN, CN, MP, M, AM, BM, CM, IDIMY, Y, IERROR, W)****DIMENSION
OF ARGUMENTS**

AN(N), BN(N), CN(N), AM(M), BM(M), CM(M), Y(IDIMY, N), W (see
argument list)

LATEST REVISION

November 1973

PURPOSE

BLKTTRI solves a system of linear equations of the form
 $AN(J)*X(I, J-1) + AM(I)*X(I-1, J) + (BN(J)+BM(I))*X(I, J)$
 $+ CN(J)*X(I, J+1) + CM(I)*X(I+1, J) = Y(I, J)$ for $I=1, 2, \dots, M$
and $J=1, 2, \dots, N$. I is evaluated modulo M and J modulo N ;
i.e., $X(I, 0) = X(I, N)$; $X(I, N+1) = X(I, 1)$; $X(0, J) = X(M, J)$;
 $X(M+1, J) = X(1, J)$. These equations usually result from
the discretization of separable elliptic equations.
Boundary conditions may be Dirichlet, Neumann or periodic.

ACCESS CARDS

*FORTRAN, S=ULIB, N=BLKTTRI
*COSY

USAGE

CALL BLKTTRI (IFLG, NP, N, AN, BN, CN, MP, M, AM, BM, CM, IDIMY, Y, IERROR, W)

ENTRY POINTS

BLKTRI, PROD, PRODP, CPROD, CPRODP, COMPB, IDX, PADD, PPADD

ARGUMENTS

On Input

IFLG

- = 0 Initialization only. Certain quantities which depend on NP, N, AN, BN and CN are computed and stored in the work array W.
- = 1 The quantities which were computed in the initialization are used to obtain the solution X(I,J).

Note: A call with IFLG=0 takes approximately twice as much time as a call with IFLG=1. However, the initialization does not have to be repeated unless NP, N, AN, BN or CN change.

NP

NP=1 If AN(1) and CN(N) are zero.

NP=0 If AN(1) and CN(N) are not zero which corresponds to the case of periodic boundary conditions.

N

The length of the arrays AN, BN and CN. If NP=1, then N must be of the form $2^k - 1$ where k is an integer > 1. If NP=0, then N must be of the form 2^k . (The operation count of the algorithm is proportional to $MN \log N$ and hence N should be selected less than or equal to M.)

AN, BN and CN

One dimensional arrays which contain coefficients of the linear equations.

MP

MP=1 If AM(1) and CM(M) are zero.

MP=0 If AM(1) and CM(M) are not zero which corresponds to the case of periodic boundary conditions.

M

The length of the arrays AM, BM and CM. M may be any integer greater than 1.

AM, BM and CM

One dimensional arrays which contain coefficients of the linear equations.

IDIMY

The first (or row) dimension of the two-dimensional array Y as it appears in the DIMENSION statement of the program which calls BLKTRI. This parameter is used for specifying the variable dimension of Y. (Hence IDIMY must be at least M.)

Y

A two-dimensional array which contains the right side of the linear system of equations. Y must be dimensioned at least M by N.

W

A one-dimensional work array. If NP = 1, then W must be dimensioned at least $[2(N+2)(\log_2(N+1)-1)-N+5*M+2]$. If NP = 0, then W must be dimensioned at least $[2*N*\log_2(N)+2+$ maximum of $2*N$ and $6*M$].

On Output

Y

Contains the solution X.

IERROR

- = 0 No error.
- = 1 $M < 2$.
- = 2 N is not of the form $2^K - 1$ when K is an integer > 1 .
- = 3 N is not of the form 2^K where K is an integer > 1 .
- = 4 BLKTRI was unable to compute the roots of certain polynomials.

On Output
(continued)

W

Contains intermediate values which must not be destroyed if BLKTRI is to be called again with IFLG = 1.

SPECIAL CONDITIONS

The algorithm may fail if $|BM(I)+BN(J)| < |AM(I)| + |AN(J)| + |CM(I)| + |CN(J)|$ for some I and J. Also, the algorithm will fail if the system does not have a solution. This is possible if the homogeneous system has a nontrivial solution. An example would be if $X(I,J) = \text{constant}$ satisfies the equation and boundary conditions set equal to zero.

I/O

None

COMMON BLOCKS

CBLKT

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Paul N. Swarztrauber, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

BLKTRI was written at NCAR and contains an algorithm which was developed at NCAR during the period January-June 1972.

ALGORITHM

The linear system of equations is solved for $X(I,J)$ by a modified cyclic reduction algorithm. A complete description may be found in the manuscript entitled *A Direct Method for the Discrete Solution of Separable Elliptic Equations* which has been accepted for publication in the SIAM Journal on Numerical Analysis.

SPACE REQUIRED

$7060_8 = 3632_{10}$ locations are required.

TIMING AND ACCURACY

The running time is proportional to $MN \log N$. The following table was obtained by solving the sample problem described at the end of this write up. The times are in milliseconds of 7600 time. The error is relative and the 7600 has between 14 and 15 decimal digits of accuracy.

<u>M</u>	<u>N</u>	<u>Initialization Time</u>	<u>Solution Time</u>	<u>Error</u>
15	15	28	12	7.99×10^{-14}
31	31	135	54	2.95×10^{-13}
63	63	621	259	3.63×10^{-12}
127	127	2807	1214	1.93×10^{-10}

PORTABILITY

There are no machine dependent constants.

REQUIRED RESIDENT
ROUTINES

SQRTF, CABS (Complex absolute values called from PPADD)

Application

In this section we will describe how to use BLKTRI to solve separable elliptic equations. We will assume that the equation is defined on $a < x < b$; $c < y < d$ and has the form

$$a(x)\frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x} + c(x)u + d(y)\frac{\partial^2 u}{\partial y^2} + e(y)\frac{\partial u}{\partial y} + f(y)u = g(x,y) \quad (1)$$

where $a(x)d(y) > 0$. In the following section, it is shown that certain equations which are not in this form can be put in this form. Assume that some grid (x_i, y_j) is placed on the region, with equal grid spacings of Δx and Δy .

Define

$$\begin{aligned} a_i &= a(x_i) & d_j &= d(y_j) \\ b_i &= b(x_i) & e_j &= e(y_j) \\ c_i &= c(x_i) & f_j &= f(y_j) \end{aligned} \quad (2)$$

$$\begin{aligned} g_{i,j} &= g(x_i, y_j) \\ u_{i,j} &= u(x_i, y_j) \end{aligned} \quad (3)$$

Then a finite difference approximation to (1) is

$$\begin{aligned}
 & a_i \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + b_i \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + c_i u_{i,j} \\
 & + d_j \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} + e_j \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \\
 & + f_j u_{i,j} = g_{i,j}
 \end{aligned} \tag{4}$$

Define

$$ax_i = \frac{a_i}{\Delta x^2} - \frac{b_i}{2\Delta x}$$

$$bx_i = \frac{-2a_i}{\Delta x^2} + c_i \tag{5}$$

$$cx_i = \frac{a_i}{\Delta x^2} + \frac{b_i}{2\Delta x}$$

$$ay_j = \frac{d_j}{\Delta y^2} - \frac{e_j}{2\Delta y}$$

$$by_j = -\frac{2d_j}{\Delta y^2} + f_j \tag{6}$$

$$cy_j = \frac{d_j}{\Delta y^2} + \frac{e_j}{2\Delta y}$$

$$Pu_{i,j} = ax_i u_{i-1,j} + bx_i u_{i,j} + cx_i u_{i+1,j} \tag{7}$$

Application
(continued)

$$Qu_{i,j} = ay_j u_{i,j-1} + by_j u_{i,j} + cy_j u_{i,j+1} \quad (8)$$

Then (4) can be written

$$Pu_{i,j} + Qu_{i,j} = g_{i,j} \quad (9)$$

We will now discuss the incorporation of the boundary conditions. We will consider the x direction only; however, the same analysis applies in the y direction. There are four possible combinations: Dirichlet-Dirichlet, Mixed-Dirichlet, Mixed-Mixed and periodic.

A. Dirichlet-Dirichlet

Here we require that $u(a, y_j) = p_j$ and $u(b, y_j) = q_j$.

Define

$$\Delta x = \frac{b-a}{M+1} \quad (10)$$

$$x_i = i\Delta x + a \quad i = 0, 1, 2, \dots, M+1 \quad (11)$$

at $i = 1$ equation (4) has the form

$$ax_1 u_{0,j} + bx_1 u_{1,j} + cx_1 u_{2,j} + Qu_{1,j} = g_{1,j} \quad (12)$$

The boundary condition specifies $u_{0,j} = p_j$ and hence (12) can be written

$$bx_1 u_{1,j} + cx_1 u_{2,j} + Qu_{1,j} = g_{1,j} - ax_1 p_j \quad (13)$$

Similarly at $i = M$ we obtain

$$ax_M u_{M-1,j} + bx_M u_{M,j} = g_{M,j} - cx_M q_j \quad (14)$$

Using (5), (13) and (14) we can determine the following quantities for input to BLKTRI.[†]

$$\begin{aligned}
 AM(1) &= 0 \\
 BM(1) &= bx_1 \\
 CM(1) &= cx_1 \\
 Y(1,J) &= g_{1,j} - ax_1 p_j \quad J = j = 2, 3, \dots, N-1
 \end{aligned} \tag{15}$$

$$\begin{aligned}
 AM(I) &= ax_i \\
 BM(I) &= bx_i \\
 CM(I) &= cx_i \\
 Y(I,J) &= g_{i,j}
 \end{aligned} \quad \begin{aligned}
 I = i &= 2, 3, \dots, M-1 \\
 J = j &= 2, 3, \dots, N-1
 \end{aligned} \tag{16}$$

$$\begin{aligned}
 AM(M) &= ax_M \\
 BM(M) &= bx_M \\
 CM(M) &= 0 \\
 Y(M,J) &= g_{M,j} - cx_M q_j \quad J = j = 2, 3, \dots, N-1
 \end{aligned} \tag{17}$$

See the discussion in E concerning the calculation of $Y(I,J)$ at the corners.

B. Mixed-Dirichlet

Here we require $\alpha u(a, y_j) + \frac{\partial}{\partial x} u(a, y_j) = p_j$. The Neumann boundary condition corresponds to the case $\alpha = 0$. At $x = b$ we also require that $u(b, y_j) = q_j$.

[†] Capital letters such as AM, BM, CM and Y refer to FORTRAN input arrays. Small letters such as ax_i , bx_i and cx_i refer to algebraic quantities.

Application
(continued)

Define

$$\Delta x = \frac{b-a}{M} \quad (18)$$

$$x_i = (i-1)\Delta x + a \quad i = 0, 1, 2, \dots, M+1 \quad (19)$$

Note that this definition of the grid differs from that given in A. Note also that $x_0 = a - \Delta x$ is outside $[a, b]$. This "virtual" point will be used as a computational device to center the boundary condition at $x = a$ and thereby provide the same accuracy throughout the region R.

At $i = 1$, equation (4) has the form

$$ax_1 u_{0,j} + bx_1 u_{1,j} + cx_1 u_{2,j} + Qu_{1,j} = g_{1,j} \quad (20)$$

The discrete form of the boundary condition is

$$-\frac{1}{2\Delta x} u_{0,j} + \alpha u_{1,j} + \frac{1}{2\Delta x} u_{2,j} = P_j \quad (21)$$

Eliminating $u_{0,j}$ between (20) and (21) we obtain

$$\begin{aligned} (bx_1 + ax_1 2\Delta x \alpha) u_{1,j} + (cx_1 + ax_1) u_{2,j} + Qu_{1,j} \\ = g_{1,j} + ax_1 2\Delta x p_j \end{aligned} \quad (22)$$

At $i = M$, equation (4) has the form

$$ax_M u_{M-1,j} + bx_M u_{M,j} + cx_M u_{M+1,j} + Qu_{M,j} = g_{M,j} \quad (23)$$

The Dirichlet boundary condition is

$$u_{M+1,j} = q_j \quad (24)$$

Hence (23) may be written

$$ax_{M-1,j} + bx_{M,j} + Qu_{M,j} = g_{M,j} - cx_{M,j} \quad (25)$$

Using (5), (22) and (25), we can determine the following quantities for input to BLKTRI.

$$\begin{aligned} AM(1) &= 0 \\ BM(1) &= bx_1 + ax_1 2\Delta x \\ CM(1) &= cx_1 + ax_1 \\ Y(1,J) &= g_{1,j} + ax_1 2\Delta x p_j \quad J = j = 2, 3, \dots, N-1 \end{aligned} \quad (26)$$

$$\begin{aligned} AM(I) &= ax_i \\ BM(I) &= bx_i \quad I = i = 2, 3, \dots, M-1 \\ CM(I) &= cx_i \\ Y(I,J) &= g_{i,j} \quad I = 2, \dots, M-1 ; \\ & \quad J = 2, \dots, N-1 \end{aligned} \quad (27)$$

$$\begin{aligned} AM(M) &= ax_M \\ BM(M) &= bx_M \\ CM(M) &= 0 \\ Y(M,J) &= g_{M,j} - cx_{M,j} \quad J = j = 2, 3, \dots, N-1 \end{aligned} \quad (28)$$

See the discussion in E concerning the calculation of $Y(I,J)$ at the corners.

C. Mixed-Mixed

Here we require $\alpha u(a, y_j) + \frac{\partial}{\partial x} u(a, y_j) = p_j$ and $\beta u(b, y_j) + \frac{\partial}{\partial x} u(b, y_j) = q_j$. The Neumann boundary

Application
(continued)

condition corresponds to the case $\alpha = \beta = 0$.

Define

$$\Delta x = \frac{b-a}{M-1} \quad (29)$$

$$x_i = (i-1)\Delta x + a \quad i = 0, 1, 2, \dots, M+1 \quad (30)$$

Note that this definition of the grid differs from that given in A and B. Note also that the points x_0 and x_{M+1} lie a distance Δx outside the interval $[a, b]$. These "virtual" points will be used as a computational device to center the boundary conditions at $x = a$ and $x = b$. This maintains the same order of accuracy through the region R.

Following the analysis at $i = 1$ in B, we observe that equation (22) also holds for the present case.

At $i = M$, equation (4) has the form

$$ax_M u_{M-1,j} + bx_M u_{M,j} + cx_M u_{M+1,j} + Qu_{M,j} = g_{M,j} \quad (31)$$

The discrete form of the boundary condition is

$$-\frac{1}{2\Delta x} u_{M-1,j} + \beta u_{M,j} + \frac{1}{2\Delta x} u_{M+1,j} = q_j \quad (32)$$

Eliminating $u_{M+1,j}$ between (31) and (32) we obtain

$$\begin{aligned} & (ax_M + cx_M)u_{M-1,j} + (bx_M - cx_M 2\Delta x \beta)u_{M,j} + Qu_{M,j} \\ & = g_{M,j} - cx_M 2\Delta x q_j \end{aligned} \quad (33)$$

Using (5), (22) and (33), we can determine the following quantities for input to BLKTRI.

$$\begin{aligned}
 AM(1) &= 0 \\
 BM(1) &= bx_1 + ax_1 2\Delta x\alpha \\
 CM(1) &= cx_1 + ax_1 \\
 Y(1,J) &= g_{1,j} + ax_1 2\Delta x p_j \quad J = j = 2, 3, \dots, N-1
 \end{aligned} \tag{34}$$

$$\begin{aligned}
 AM(I) &= ax_i \\
 BM(I) &= bx_i \quad I = i = 2, 3, \dots, M-1 \\
 CM(I) &= cx_i \\
 Y(I,J) &= g_{i,j} \quad I = 2, \dots, M-1 ; \\
 & \quad J = 2, \dots, N-1
 \end{aligned} \tag{35}$$

$$\begin{aligned}
 AM(M) &= ax_M + cx_M \\
 BM(M) &= bx_M - cx_M 2\Delta x\beta \\
 CM(M) &= 0 \\
 Y(M,J) &= g_{M,j} - cx_M 2\Delta x q_j \quad J = j = 2, 3, \dots, N-1
 \end{aligned} \tag{36}$$

See the discussion in E concerning the calculation of $Y(I,J)$ at corners.

D. Periodic

Here we require that $u(a+x, y_j) = u(b+x, y_j)$ for all x .

Define

$$\Delta x = \frac{b-a}{M} \tag{37}$$

Application
(continued)

$$x_i = (i-1)\Delta x + a \quad i = 0,1,2,\dots,M+1 \quad (38)$$

At $i = 1$, equation (4) has the form (20). But from the boundary condition we have

$$u_{0,j} = u_{M,j} \quad (39)$$

Therefore (20) has the form

$$ax_{1M,j} + bx_{11,j} + cx_{12,j} + Qu_{1,j} = g_{1,j} \quad (40)$$

Similarly at $i = M$, we use (31) and the boundary condition $u_{M+1} = u_{1,j}$ to obtain

$$ax_{MM-1,j} + bx_{MM,j} + cx_{M1,j} + Qu_{M,j} = g_{M,j} \quad (41)$$

Using (5), (40) and (41) we can determine the following quantities for input to BLKTRI.

$$\begin{aligned} AM(I) &= ax_i \\ BM(I) &= bx_i \\ CM(I) &= cx_i \end{aligned} \quad I = i = 1,2,\dots,M \quad (42)$$

Note that since $AM(1)$ and $CM(M)$ are not zero we must set $MP = 0$ in the calling sequence.

$$Y(I,J) = g_{i,j} \quad \begin{array}{l} I = 1,2,\dots,M \\ J = 2,\dots,N-1 \end{array} ; \quad (43)$$

E. Corners

For $I = 2, \dots, M-1$ and $J = 2, \dots, N-1$, we set $Y(I,J) = g_{i,j}$. However, on the edges $Y(I,J)$ will be modified as a result of the boundary conditions (except for the periodic case). For example, this modification appears in the last of equations (15). We note however, that in (15) the indices $J = 1$ and $J = N$ do not appear. This corresponds to a corner and here the modification will include contributions from the two boundary conditions which meet at the corner. Consider the case of $Y(1,1)$ where Dirichlet boundary conditions $u(a,y_j) = p_j$ and $u(x_i,c) = q_i$ meet.

Then equation (4) centered at $i = 1, j = 1$ is

$$\begin{aligned} ax_1 u_{0,1} + bx_1 u_{1,1} + cx_1 u_{2,1} + ay_1 u_{1,0} + by_1 u_{1,1} \\ + cy_1 u_{1,2} = g_{1,1} \end{aligned} \quad (44)$$

But $u_{0,1} = p_1$ and $u_{1,0} = q_1$. Therefore (44) has the form

$$\begin{aligned} bx_1 u_{1,1} + cx_1 u_{2,1} + by_1 u_{1,1} + cy_1 u_{1,2} \\ = g_{1,1} - ax_1 p_1 - ay_1 q_1 \end{aligned} \quad (45)$$

Hence the input to BLKTRI is

$$Y(1,1) = g_{1,1} - ax_1 p_1 - ay_1 q_1 \quad (46)$$

Sample Problem

In this section we will demonstrate the use of BLKTRI in obtaining a discrete solution to Poissons equation inside a sphere and subject to Dirichlet boundary conditions at $r = r_0$. We will assume equatorial symmetry and hence limit our attention to the northern hemisphere. Further, we will assume axisymmetry, (i.e., solution is constant as a function of longitude). This reduces the problem computationally to 2 dimensions. Hence we wish to approximate a solution $u(r,\theta)$ of the equation

$$\frac{\partial^2 u}{\partial r^2} + \frac{2}{r} \frac{\partial u}{\partial r} + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2} + \frac{\cos \theta}{r^2 \sin \theta} \frac{\partial u}{\partial \theta} = g(r,\theta) \quad \begin{matrix} 0 \leq r \leq r_0 \\ 0 \leq \theta \leq \pi/2 \end{matrix} \quad (47)$$

subject to the boundary conditions

$$u(r_0, \theta) = h(\theta) \quad 0 \leq \theta \leq \pi/2 \quad (48)$$

and for $0 < r \leq r_0$

$$\frac{\partial}{\partial \theta} u(r, \pi/2) = 0 \quad (\text{equatorial symmetry}) \quad (49)$$

$$\frac{\partial}{\partial \theta} u(r, 0) = 0 \quad (\text{axisymmetry}) \quad (50)$$

We can put (47) in the form (1) by multiplying by r^2

$$r^2 \frac{\partial^2 u}{\partial r^2} + 2r \frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial \theta^2} + \cot \theta \frac{\partial u}{\partial \theta} = r^2 g(r,\theta) \quad (51)$$

Now for $\theta = 0$ the term $\cot \theta \frac{\partial u}{\partial \theta}$ is indeterminate. However, from (50) and L'Hospital's rule we determine its value as $\frac{\partial^2 u}{\partial \theta^2}$. Hence at $\theta = 0$ we must have

$$r^2 \frac{\partial^2 u}{\partial r^2} + 2r \frac{\partial u}{\partial r} + 2 \frac{\partial^2 u}{\partial \theta^2} = r^2 g(r,\theta) \quad \text{at } \theta = 0 \quad (52)$$

Select integers M, N and define

$$\Delta\theta = \frac{\pi}{2(M-1)} \quad (53)$$

$$\theta_i = (i-1)\Delta\theta \quad i = 0, 1, 2, \dots, M+1 \quad (54)$$

$$\Delta r = \frac{r_0}{N+1}$$

$$r_j = j\Delta r \quad j = 0, 1, 2, \dots, N+1 \quad (55)$$

Then the finite difference form of (51) is

$$\begin{aligned} & j^2(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) + j(u_{i,j+1} - u_{i,j-1}) \\ & + \frac{1}{\Delta\theta^2} (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) \\ & + \frac{\cot\theta_i}{2\Delta\theta} (u_{i+1,j} - u_{i-1,j}) = j^2\Delta r^2 g_{i,j} \end{aligned} \quad (56)$$

where $u_{i,j} \approx u(r_i, \theta_i)$ and $g_{i,j} = g(r_i, \theta_i)$.

Define

$$\begin{aligned} ar_j &= j(j-1) \\ br_j &= -2j^2 \\ cr_j &= j(j+1) \end{aligned} \quad (57)$$

Sample Problem
(continued)

$$a_{\theta_i} = \frac{1}{\Delta\theta^2} - \frac{\cot\theta_i}{2\Delta\theta}$$

$$b_{\theta_i} = -\frac{2}{\Delta\theta^2} \quad (58)$$

$$c_{\theta_i} = \frac{1}{\Delta\theta^2} + \frac{\cot\theta_i}{2\Delta\theta}$$

$$\hat{g}_{i,j} = j^2 \Delta r^2 g_{i,j} \quad (59)$$

$$P u_{i,j} = a_{\theta_i} u_{i-1,j} + b_{\theta_i} u_{i,j} + c_{\theta_i} u_{i+1,j} \quad (60)$$

$$Q u_{i,j} = a_{r_j} u_{i,j-1} + b_{r_j} u_{i,j} + c_{r_j} u_{i,j+1} \quad (61)$$

Then (56) has the form

$$P u_{i,j} + Q u_{i,j} = \hat{g}_{i,j} \quad (62)$$

We will now develop the discretization in the θ direction.

The finite difference form of (52) is

$$Q u_{1,j} + \frac{2}{\Delta\theta^2} (u_{0,j} - 2u_{1,j} + u_{2,j}) = \hat{g}_{1,j} \quad (63)$$

However the discrete form of (50) is $u_{0,j} = u_{2,j}$ which substituted into (63) yields

$$Q u_{1,j} - \frac{4}{\Delta\theta^2} u_{1,j} + \frac{4}{\Delta\theta^2} u_{2,j} = \hat{g}_{1,j} \quad (64)$$

At the equator $\theta = \pi/2$, $i = M$ and (56) has the form

$$Qu_{M,j} + \frac{1}{\Delta\theta^2} (u_{M+1,j} - 2u_{M,j} + u_{M-1,j}) = \hat{g}_{M,j} \quad (65)$$

The discrete form of (49) is $u_{M+1,j} = u_{M-1,j}$ which when substituted into (65) yields

$$Qu_{M,j} + \frac{2}{\Delta\theta^2} u_{M-1,j} - \frac{2}{\Delta\theta^2} u_{M,j} = \hat{g}_{M,j} \quad (66)$$

Using (58), (64) and (66) we can determine the following quantities for input to BLKTRI.

$$AM(1) = 0$$

$$BM(1) = -\frac{4}{\Delta\theta^2} \quad (67)$$

$$CM(1) = \frac{4}{\Delta\theta^2}$$

$$AM(I) = \frac{1}{\Delta\theta^2} - \frac{\cot\theta_i}{2\Delta\theta}$$

$$BM(I) = -\frac{2}{\Delta\theta^2} \quad I = i = 2, 3, \dots, M-1 \quad (68)$$

$$CM(I) = \frac{1}{\Delta\theta^2} + \frac{\cot\theta_i}{2\Delta\theta}$$

$$AM(M) = \frac{2}{\Delta\theta^2}$$

$$BM(M) = -\frac{2}{\Delta\theta^2} \quad (69)$$

$$CM(M) = 0$$

Sample Problem
(continued)

We will now develop the discretization in the r direction.

At $j = 1$ (56) has the form

$$-2u_{i,1} + 2u_{i,2} + Pu_{i,1} = \hat{g}_{i,1} \quad (70)$$

where we have the pleasant result that the coefficient of $u_{1,0}$ is zero. Hence the system of equations does not include $u_{0,0}$ which is the value of the solution at the center of the sphere. However, once the solution is obtained elsewhere then $u_{0,0}$ may be determined using an integral form of (47). See (76).

At $j = N$ (56) has the form

$$\begin{aligned} N(N-1)u_{i,N-1} - 2N^2u_{i,N} + N(N+1)u_{i,N+1} \\ + Pu_{i,N} = \hat{g}_{i,N} \end{aligned} \quad (71)$$

From the discrete form of (48) we have $u_{i,N+1} = h_i$ hence (71) can be written

$$N(N-1)u_{i,N-1} - 2N^2u_{i,N} + Pu_{i,N} = \hat{g}_{i,N} - N(N+1)h_i \quad (72)$$

Using (57), (70) and (72) we can determine the following quantities for input to BLKTRI.

$$\begin{aligned} AN(1) &= 0 \\ BN(1) &= -2 \\ CN(1) &= 2 \end{aligned} \quad (73)$$

$$\begin{aligned} AN(J) &= J(J-1) \\ BN(J) &= -2J^2 \\ CN(J) &= J(J+1) \end{aligned} \quad J = 2, 3, \dots, N-1 \quad (74)$$

$$\begin{aligned}
 AN(N) &= N(N-1) \\
 BN(N) &= -2N^2 \\
 CN(N) &= 0
 \end{aligned} \tag{75}$$

$$\begin{aligned}
 Y(I,J) &= \hat{g}_{i,j} & I &= 1,2,\dots,M; \\
 & & J &= 1,2,\dots,N-1
 \end{aligned}$$

$$Y(I,N) = \hat{g}_{i,N} - N(N+1)h_i \quad I = 1,2,\dots,M$$

After the call to BLKTRI then it remains to determine $u_{0,0}$ which is the solution at the center of the sphere. If we multiply (47) by $r^2 \sin\theta$ and then integrate, we obtain

$$\int_0^{\pi/2} \sin\theta \frac{\Delta r^2}{4} \frac{\partial u}{\partial r} \Big|_{\frac{\Delta r}{2}} d\theta = \int_0^{\pi/2} \int_0^{\frac{\Delta r}{2}} r^2 \sin\theta g(r,\theta) dr d\theta \tag{76}$$

We then make the approximations $\frac{\partial u}{\partial r} \Big|_{\frac{\Delta r}{2}} \approx \frac{u_{i,1} - u_{0,0}}{\Delta r}$ and $g(r,\theta) \approx g_{0,0}$ and obtain

$$\int_0^{\pi/2} \sin\theta \frac{\Delta r^2}{4} \frac{u_{i,1} - u_{0,0}}{\Delta r} d\theta = \int_0^{\pi/2} \sin\theta g_{0,0} \frac{\Delta r^3}{24} d\theta \tag{77}$$

or

$$\int_0^{\pi/2} \sin\theta u_{i,1} d\theta - u_{0,0} = \frac{\Delta r^2}{6} g_{0,0} \tag{78}$$

Finally we obtain an expression for $u_{0,0}$

$$u_{0,0} = \frac{\Delta\theta}{2} [y_{M,1} + 2 \sum_{i=2}^{M-1} \sin\theta_i u_{i,1}] - \frac{\Delta r^2}{6} g_{0,0} \tag{79}$$

SUBROUTINE POIS (IFLAG,NPEROD,N,MPEROD,M,A,B,C,IDIMY,Y,W)**DIMENSION OF ARGUMENTS**

A(M),B(M),C(M),Y(IDIMY,N),W(5.25*N+5*M)

LATEST REVISION

August 20, 1973

PURPOSE

POIS solves the standard finite difference approximation to Poisson's equation.

ACCESS CARDS*FORTRAN,S=ULIB,N=POIS
*COSY**USAGE**

CALL POIS (IFLAG,NPEROD,N,MPEROD,M,A,B,C,IDIMY,Y,W)

ARGUMENTS**On Input****IFLAG**

- = 0 if this call is the initial entry to POIS with a given value of N and NPEROD.
- = 1 if N and NPEROD are unchanged from previous call to POIS.

NPEROD

- Indicates the boundary conditions assumed in the y-direction.
- = 0 if periodic.
- = 1 if Dirichlet on both sides.
- = 2 if Dirichlet on left and Neumann on right side.
- = 3 if Neumann on both sides.

On Input
(continued)

N

Is the number of unknowns in the y-direction. N is dependent on the boundary conditions and must have the following form:

NPEROD

$$= 0 \text{ or } 2, \text{ then } N = 2^P 3^Q 5^R$$

$$= 1, \text{ then } N = 2^P 3^Q 5^R - 1$$

$$= 3, \text{ then } N = 2^P 3^Q 5^R + 1$$

where P, Q and R may be any non-negative integers. N must be greater than 2.

MPEROD

= 0 if there are periodic boundary conditions in the x-direction.

= 1 for all other boundary conditions.

M

The number of unknowns in the x-direction. M is dependent on the boundary conditions. The effect is illustrated in a section below.

A,B,C

Coefficients of the finite difference equations, i.e., $A(I)*V(I-1,J) + B(I)*V(I,J) + C(I)*V(I+1,J) + V(I,J-1) - 2*V(I,J) + V(I,J+1) = Y(I,J)$. The V array is the finite difference approximation.

IDIMY

Row dimension of the array Y in the calling program. This parameter is used for specifying the variable dimension.

Y

The right side of the equation.

W

An array which must be provided for work space. W must be dimensioned at least $5.25*N + 5*M$.

SPACE REQUIRED

6027₈ = (3095₁₀) locations

In the subroutine POPRNT a call is made to the I/O output package. If space is critical this subroutine can be deleted or modified to omit printing the message as it is now written.

ACCURACY

At least 10 digits for most cases.

This subroutine produces a very accurate solution. The number of significant digits in the answer decreases as m and n increases, but for any reasonable problem one can expect at least ten digits.

There are two cases when the system of equations may be singular, i.e., when only Neumann or periodic conditions are imposed on the boundary of the rectangle. In these cases a solution may not exist. A point is reached when the single equation $\alpha z = \beta$ must be solved. If $\alpha \neq 0$, then there is no problem. However, if $\alpha = 0$, then a solution exists if and only if $\beta = 0$. The program assumes that $\beta = 0$, but prints out α and β with a message to this effect. If, in fact, $\beta \neq 0$ the solution will be incorrect.

TIMING

The approximate running time for a 63 x 63 grid on the NCAR CDC 7600 is 137 milliseconds. In general, the running time is proportional to $N \log_2 N$.

PORTABILITY

There are three machine dependent constants located at the beginning of POINIT, TRID, and TRIDP.

REQUIRED RESIDENT ROUTINES

COS

Method

Subroutine POIS is used to solve a finite difference approximation to certain elliptic partial differential equations. Suppose we want to solve the equation

$$r(x) \frac{\partial^2 u}{\partial x^2} + s(x) \frac{\partial u}{\partial x} + t(x)u + \frac{\partial^2 u}{\partial y^2} = f(x,y) \quad , \quad (1)$$

on the rectangle, $a < x < b$, $c < y < d$, with certain boundary conditions imposed upon the solution u along the four boundaries. We define a grid with spacings Δx and Δy and approximate all derivatives of (1) by finite differences to obtain the equation

$$r_i \frac{v_{i-1,j} - 2v_{i,j} + v_{i+1,j}}{\Delta x^2} + s_i \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} + t_i v_{i,j} + \frac{v_{i,j-1} - 2v_{i,j} + v_{i,j+1}}{\Delta y^2} = f_{i,j} \quad , \quad (2)$$

where $r_i = r(x_i)$, $s_i = s(x_i)$, $t_i = t(x_i)$, $f_{i,j} = f(x_i, y_j)$, and $v_{i,j}$ is an approximation to $u(x_i, y_j)$. This is the basic form of the equation which is used to solve for the unknowns $v_{i,j}$, $i = 1, 2, \dots, M$, $j = 1, 2, \dots, N$. The above equation must be modified at the boundaries to reflect the effect of the given boundary conditions. By defining

$$a_i = r_i \frac{\Delta y^2}{\Delta x^2} - s_i \frac{\Delta y^2}{2\Delta x}$$

$$b_i = -2r_i \frac{\Delta y^2}{\Delta x^2} + t_i \Delta y^2$$

$$c_i = r_i \frac{\Delta y^2}{\Delta x^2} + s_i \frac{\Delta y^2}{2\Delta x}$$

Method
(continued)

$$y_{i,j} = \Delta y^2 f_{i,j}$$

$$Qv_{i,j} = v_{i,j-1} - 2v_{i,j} + v_{i,j+1}$$

equation (2) may be re-written in the form

$$a_i v_{i-1,j} + b_i v_{i,j} + c_i v_{i+1,j} + Qv_{i,j} = y_{i,j} \quad (3)$$

**Incorporating
Boundary Data**

Two sets of boundary conditions are specified: one for the x-direction and one for the y-direction. We will discuss in detail the effect of boundary conditions in the x-direction.

Dirichlet-Mixed: Assume that the boundary conditions are

$$u(a, y_j) = p_j \quad (4)$$

and

$$\alpha u(b, y_j) + \frac{\partial u}{\partial x}(b, y_j) = q_j \quad (5)$$

where α is a constant. (Note that if $\alpha = 0$, we have the Neuman boundary condition $\frac{\partial u}{\partial x}(b, y_j) = q_j$.) For this case we define $\Delta x = \frac{b-a}{M}$ and

$$x_i = a + i\Delta x, \quad i = 0, 1, 2, \dots, M.$$

We see that the unknowns for a given y_j are $v_{i,j}$ for $i=1, 2, \dots, M$. For $i=2, 3, \dots, M-1$, the unknowns satisfy equation (3). For $i=1$, equations (3) and (4) combine to

$$b_1 v_{1,j} + c_1 v_{1,j} + Qv_{1,j} = y_{1,j} - a_1 p_j \quad (6)$$

For $i = M$, we first approximate the boundary condition (5) by

$$\alpha v_{M,j} + \frac{v_{M+1,j} - v_{M-1,j}}{2\Delta x} = q_j \quad (7)$$

We solve this for $v_{M+1,j}$ and substitute it for $v_{M+1,j}$ in (3) to get

$$\begin{aligned} (a_M + c_M) v_{M-1,j} + (b_M - 2\alpha\Delta x c_M) v_{M,j} + Qv_{1,j} \\ = y_{M,j} - 2\Delta x q_j \end{aligned} \quad (8)$$

From (3), (6) and (8) we determine the quantities for input to POIS as:¹

$$\begin{aligned} \text{MPEROD} &= 1 \\ A(1) &= 0 \\ B(1) &= b_1 \\ C(1) &= c_1 \\ Y(1,J) &= y_{1,j} - a_1 p_1, \quad J = j = 2, 3, \dots, N-1 \end{aligned}$$

$$\left. \begin{aligned} A(I) &= a_i \\ B(I) &= b_i \\ C(I) &= c_i \\ Y(I,J) &= y_{i,j}, \quad J = j = 2, 3, \dots, N-1 \end{aligned} \right\} I = i = 2, 3, \dots, M-1 \quad (9)$$

$$\begin{aligned} A(M) &= a_M + c_M \\ B(M) &= b_M - 2\alpha\Delta x c_M \\ C(M) &= 0 \\ Y(M,J) &= y_{M,j} - 2\Delta x q_j, \quad J = j = 2, 3, \dots, N-1 \end{aligned}$$

See the discussion in "Corners" for the values of $Y(I,J)$ at corners.

¹ Capital letters such as A, B, C and Y refer to FORTRAN input arrays. Small letters such as a_i , b_i , c_i and $y_{i,j}$ refer to algebraic quantities.

Incorporating
Boundary Data
(continued)

Mixed-Dirichlet: Suppose that the boundary conditions are

$$\alpha u(a, y_j) + \frac{\partial u}{\partial x}(a, y_j) = p_j \quad (10)$$

and

$$u(b, y_j) = q_j \quad (11)$$

For this case we define $\Delta x = \frac{b-a}{M}$ and

$$x_i = a + (i-1)\Delta x, \quad i = 1, 2, 3, \dots, M+1 \quad .$$

We see that the unknowns for a given y_j in this case are still $v_{i,j}$ for $i = 1, 2, \dots, M$. (However, note that in this case the subscript i refers to a different x value than in the previous case.)

For $i = 2, 3, \dots, M-1$, the unknowns satisfy equation (3).

For $i = 1$, we discretize equation (10) to get

$$\alpha v_{1,j} + \frac{v_{2,j} - v_{0,j}}{2\Delta x} = p_j \quad (12)$$

and solve it for $v_{0,j}$. Substituting this value for $v_{0,j}$ in equation (3) yields

$$\begin{aligned} (b_1 + 2\Delta x \alpha a_1) v_{1,j} + (c_1 + a_1) v_{2,j} + Q v_{1,j} \\ = y_{1,j} + 2\Delta x a_1 p_j \quad . \end{aligned} \quad (13)$$

For $i = M$, equation (3) becomes

$$a_M v_{M-1,j} + b_M v_{M,j} + Q_{v_{M,j}} = y_{M,j} - c_M q_j \quad (14)$$

From (3), (13), and (14) we determine the quantities for input to POIS as

$$\begin{aligned} \text{MPEROD} &= 1 \\ A(1) &= 0. \\ B(1) &= b_1 + 2\Delta x \alpha a_1 \\ C(1) &= c_1 + a_1 \\ Y(1,J) &= y_{1,j} + 2\Delta x a_1 p_j, \quad J = j = 2, 3, \dots, N-1 \end{aligned}$$

$$\begin{aligned} A(M) &= a_M \\ B(M) &= b_M \\ C(M) &= 0. \\ Y(M,J) &= y_{M,j} - c_M q_j, \quad J = j = 2, 3, \dots, N-1 \end{aligned}$$

and the quantities in equation (9).

See the discussion in "Corners" for the values of $Y(I,J)$ at corners.

Periodic: Suppose now that the boundary conditions are given as

$$u(b+x, y_j) = u(a+x, y_j) \text{ for all } x. \quad (15)$$

For this case we define $\Delta x = \frac{b-a}{M}$ and

$$x_i = a + (i-1)\Delta x, \quad i = 0, 1, 2, \dots, M+1.$$

For a given y_j the unknowns $v_{i,j}$, $i = 2, 3, \dots, M-1$, satisfy equation (3). From (15) we get that

$$v_{0,j} = v_{M,j} \text{ and } v_{M+1,j} = v_{1,j}.$$

Incorporating
Boundary Data
(continued)

Periodic (continued): For $i = 1$, equation (3) becomes

$$a_1 v_{1M,j} + b_1 v_{11,j} + c_1 v_{12,j} + Q_{u1,j} = y_{1,j} \quad (1)$$

and for $i = M$, it becomes

$$a_M v_{M-1,j} + b_M v_{MM,j} + c_M v_{M1,j} + Q_{vM,j} = y_{M,j} \quad (1)$$

From (3), (16), and (17) we determine the quantities for input to POIS as

$$MPEMOD = 0$$

$$\left. \begin{array}{l} A(I) = a_i \\ B(I) = b_i \\ C(I) = c_i \\ Y(I,J) = y_{i,j}, \quad J = j = 1, 2, \dots, N \end{array} \right\} I = i = 1, 2, 3, \dots, M$$

Boundary Conditions in y-direction: For the boundary conditions in the y-direction the modifications to equation (3) proceed just as above for the x-direction. In this case there are no modifications necessary to the arrays A(I), B(I), and C(I), only to columns 1 and N of the input array. The proper form of $Q_{v_{i,j}}$ near the boundaries is determined in the program from the input parameter NPEMOD.

Corners: The user must be careful at the corners to ensure that boundary data (if present) enters from both boundaries. In general this will affect the values of $Y(1,1)$, $Y(1,N)$, $Y(M,1)$, and $Y(M,N)$.

For instance suppose that Dirichlet boundary conditions are given at $x = (x_0) = a$ and $y = (y_0) = c$. In this case we have $u(a, y_j) = p_j$ and $u(x_i, c) = q_i$. For $i = j = 1$ equation (3) becomes

$$a_1 v_{0,1} + b_1 v_{1,1} + c_1 v_{2,1} + v_{1,0} + 2v_{1,1} + v_{1,2} = y_{1,1} \quad .$$

Using the boundary data we get

$$b_1 v_{1,1} + c_1 v_{2,1} - 2v_{1,1} + v_{1,2} = y_{1,1} - a_1 p_1 - q_1 \quad .$$

Hence, $Y(1,1)$ should be defined as

$$Y(1,1) = y_{1,1} - a_1 p_1 - q_1 \quad .$$

Similar treatments are necessary for $Y(1,N)$, $Y(M,1)$, and $Y(M,N)$.

Example

Suppose we wish to approximate the solution of the equation

$$(3x^2+1) \frac{\partial^2 u}{\partial x^2} + \cos x \frac{\partial u}{\partial x} - 7u + \frac{\partial^2 u}{\partial y^2} = \sin(x+y) \quad (18)$$

on the rectangle $0 < x < 1$, $0 < y < 1$, with the boundary conditions

$$\left. \begin{array}{l} u(0,y) = 1+2y \\ u(1,y) = 2+\sin\pi y \\ u(x,0) = 1+x \\ \frac{\partial u}{\partial y}(x,1) = \cos\pi x \end{array} \right\} \begin{array}{l} 0 \leq y \leq 1 \\ 0 \leq x \leq 1 \end{array}$$

Example
(continued)

Since for a given y_j , $u(0, y_j)$ is known, from "Dirichlet-Mixed" above the first unknown will be $u(\Delta x, y_j)$. Also $u(1, y_j)$ is known, hence from "Mixed-Dirichlet" above we see that the last unknown will be $u(1-\Delta x, y_j)$. Hence we define $\Delta x = \frac{1}{M+1}$ and

$$x_i = i\Delta x, \quad i = 0, 1, 2, \dots, M+1 .$$

Then the unknowns are $u(x_i, y_j)$ for $i = 1, 2, \dots, M$.

For a given x_i , $u(x_i, 0)$ is known while from "Dirichlet-Mixed" above (applied to y variable) $u(x_i, 1)$ is unknown. Hence, we define $\Delta y = \frac{1}{N}$,

$$y_j = j\Delta y \quad , \quad j = 0, 1, 2, \dots, N \quad ,$$

and the unknowns are $u(x_i, y_j)$, $j = 1, 2, \dots, N$. Furthermore, N must have the form $2^p 3^q 5^r$.

From the boundary conditions we have

$$\left. \begin{aligned} v_{0,j} &= 1 + 2j\Delta y \\ v_{M+1,j} &= 2 + \sin(\pi j\Delta y) \end{aligned} \right\} \begin{array}{l} i = 0, 1, 2, \dots, N, \\ \end{array} \quad \begin{array}{l} (19) \\ (20) \end{array}$$

$$\left. \begin{aligned} v_{i,0} &= 1 + i\Delta x \\ v_{i,N+1} &= v_{i,N-1} + 2x \cos(\pi i\Delta x) \end{aligned} \right\} \begin{array}{l} i = 0, 1, 2, \dots, M+1 . \\ \end{array} \quad \begin{array}{l} (21) \\ (22) \end{array}$$

The finite difference approximation (3) for the equation (18) has the coefficients

$$a_i = (1+3i^2\Delta x^2) \left(\frac{\Delta y}{\Delta x} \right)^2 - \cos(i\Delta x) \left(\frac{\Delta y^2}{2\Delta x} \right)$$

$$b_i = -2(1+3i^2\Delta x^2) \left(\frac{\Delta y}{\Delta x} \right)^2 - 7\Delta y^2$$

$$c_i = (1+3i^2\Delta x^2) \left(\frac{\Delta y}{\Delta x} \right) + \cos(i\Delta x) \left(\frac{\Delta y^2}{2\Delta x} \right)$$

and the right side

$$y_{i,j} = \Delta y^2 \sin(i\Delta x + j\Delta y) .$$

Hence, we input into POIS the following data:

$$\text{MPEROD} = 1$$

$$\text{NPEROD} = 2$$

$$A(1) = 0.$$

$$B(1) = b_1$$

$$C(1) = c_1$$

$$Y(1,J) = y_{1,j} - a_1 v_{0,j} , J = j = 2, 3, \dots, N-1$$

$$A(I) = a_i$$

$$B(I) = b_i$$

$$C(I) = c_i$$

$$Y(I,J) = y_{i,j} , J = j = 2, 3, \dots, N-1$$

$$\left. \begin{array}{l} A(I) = a_i \\ B(I) = b_i \\ C(I) = c_i \\ Y(I,J) = y_{i,j} , J = j = 2, 3, \dots, N-1 \end{array} \right\} I = i = 2, 3, \dots, M-1$$

$$A(M) = a_M$$

$$B(M) = b_M$$

$$C(M) = 0.$$

$$Y(M,J) = y_{M,j} - c_M v_{M+1,j} , J = j = 2, 3, \dots, N-1$$

and the corner points

$$Y(1,1) = y_{1,1} - a_1 v_{0,1} - v_{1,0}$$

$$Y(M,1) = y_{M,1} - c_M v_{M+1,1} - v_{M,0}$$

Example*(continued)*

$$Y(1,N) = y_{1,N} - a_1 v_{0,N} - 2\Delta x \cos(\pi\Delta x)$$

$$Y(M,N) = y_{M,N} - c_M v_{M+1,1} - 2\Delta x \cos(\pi M \Delta x)$$

and the two y boundaries

$$\left. \begin{aligned} Y(I,1) &= y_{i,1} - v_{i,0} \\ Y(I,N) &= y_{i,N} - 2\Delta x \cos(\pi i \Delta x) \end{aligned} \right\} I = i = 2, 3, \dots, M-1 .$$

On Input
(continued)

M

The number of panels into which the interval $[A,B]$ is to be subdivided. (Hence, there will be $M + 1$ grid points in the x -direction.)

MBDCND

Indicates the boundary conditions which are assumed at the two boundaries, $x = A$ and $x = B$.

- = 0 if the solution is periodic in x .
- = 1 if the solution is specified at both boundaries.
- = 2 if the solution is specified at $x = A$ and the normal derivative of the solution is specified at $x = B$.
- = 3 if the normal derivative is specified at both boundaries.
- = 4 if the normal derivative is specified at $x = A$ and the solution is specified at $x = B$.

BDA,BDB

Arrays of dimension $N + 1$ which give the values of the normal derivatives of the solution at $x = A$ and $x = B$ respectively, when these boundary conditions are assumed.

C,D

The range of the y -variable, i.e., $C \leq y \leq D$. C must be less than D .

N

The number of panels into which the interval $[C,D]$ is to be subdivided. (Hence, there will be $N + 1$ grid points in the y -direction.) N must be of the form $2^p 3^q 5^r$ where p , q , and r are any non-negative integers. N must be greater than 2.

NBDCND

Indicates the boundary conditions assumed at $y = C$ and $y = D$ just as for the x -boundaries.

BDC,BDD

Arrays of dimension $M + 1$ which give the values of the normal derivative at $y = C$ and $y = D$, respectively, when these boundary conditions are assumed.

ELMBDA

The constant λ in the Helmholtz equation. If $\lambda > 0$, a solution may not exist.

F

A two-dimensional array giving the values of the forcing function f and the values of the solution on the boundaries (when specified). F must be dimensioned at least $(M + 1) \times (N + 1)$.

IDIMF

The row (first) dimension of the array F in the calling program. This parameter is used for specifying the variable dimension of F . (Note that $IDIMF$ must be at least $M + 1$.)

W

An array which must be provided for work space. W must be dimensioned at least $5.25*N+8*M$.

On Output

F

Contains the solution of the finite difference approximation to the problem.

PERTRB

If one specifies periodic or normal derivative boundary conditions for a Poisson equation ($\lambda = 0$), the approximation is a singular system. Hence, a solution may not exist. $PERTRB$ is a constant which is calculated and subtracted from the forcing function to ensure that a solution exists.

On Output
(continued)

IERROR

An error flag which indicates invalid parameters. The program is not executed, except for #6.

- = 0 no error.
- = 1 if $A \geq B$.
- = 2 if $MBDCND < 0$ or $MBDCND > 4$
- = 3 if $C \geq D$.
- = 4 if $N \neq 2^p 3^q 5^r$ or $N \leq 2$.
- = 5 if $NBDCND < 0$ or $NBDCND > 4$.
- = 6 if $\lambda > 0$.

Since this is the only means of indicating a possibly incorrect call to PWSCRT, the user should test this error flag after the call.

W

Contains intermediate values which must not be destroyed if PWSCRT will be called again with INTL = 1.

ENTRY POINTS

PWSCRT, NCHECK, POIS, POISGN, POINTT, TRIDP, TRID

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Roland Sweet, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized September 1, 1973

ALGORITHM

The linear system of equations is solved by cyclic reduction using the subroutine POIS. A complete description of the algorithm may be found in "A Generalized Cyclic Reduction Algorithm," by R. Sweet, SIAM J. on Numerical Analysis (to appear).

SPACE REQUIRED

$(7200)_8 = 3712_{10}$ locations are required.

ACCURACY

At least 10 digits for most cases.

This subroutine produces a very accurate solution. The number of significant digits in the answer decreases as M and N increase, but for any reasonable problem one should expect at least ten digits.

TIMING

The running time is proportional to $MN \log_2 N$. A 65×65 grid ($M = N = 64$) with $MBDCND = NBDCND = 1$ takes 144 milliseconds on the NCAR CDC 7600. (Subroutine POIS takes 137 milliseconds so the set-up time is negligible.)

PORTABILITY

There are three machine dependent constants located at the beginning of POINTT, TRID, AND TRIDP.

REQUIRED RESIDENT ROUTINES

COS

Method

Subroutine PWSCRT is used to solve the standard five-point finite-difference approximation to Helmholtz's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \lambda u = f(x,y) \quad (1)$$

on the rectangle $a < x < b$, $c < y < d$ with a wide variety of boundary conditions. In the parameter list of PWSCRT the quantities A, B, C, D, and ELMBDA represent a, b, c, d, and λ , respectively.

Method*(continued)*

We define a grid of points (x_i, y_j) by selecting integers M and N and defining

$$x_i = a + (i-1)\Delta x \quad , \quad i = 1, 2, \dots, M+1 \quad (2)$$

$$y_j = c + (j-1)\Delta y \quad , \quad j = 1, 2, \dots, N+1 \quad ,$$

where $\Delta x = \frac{b-a}{M}$ and $\Delta y = \frac{d-c}{N}$. The integers M and N appear in the parameter list also. If we now approximate each derivative of (1) by the usual second-order central difference and denote the approximation to $u(x_i, y_j)$ by v_{ij} , we get

$$\frac{v_{i-1,j} - 2v_{ij} + v_{i+1,j}}{\Delta x^2} + \frac{v_{i,j-1} - 2v_{ij} + v_{i,j+1}}{\Delta y^2} + \lambda v_{ij} = f(x_i, y_j) \quad (3)$$

The array F in the parameter list must be dimensioned at least $(M+1) \times (N+1)$. The entries of F for $I = 2, 3, \dots, M$; $J = 2, 3, \dots, N$ are

$$F(I, J) = f(x_I, y_J) \quad (4)$$

Equation (3) is the basic equation used to determine the v_{ij} . Near the boundaries this equation is modified by the boundary conditions. The three types of boundary conditions which may be assumed are (say, at $x = a$):

Solution Specified: We then are given a function g such that

$$u(a, y) = g(y), \quad c \leq y \leq d \quad .$$

In this case we set

$$v_{1,j} = g(y_j) , j = 1,2,\dots,N+1 . \quad (5)$$

The first row of the array F is

$$F(1,J) = g(y_J) , J = 1,2,\dots,N+1 .$$

Normal Derivative of the Solution Specified: We then are given a function $h(y)$ such that

$$\frac{\partial u}{\partial x}(a,y) = h(y) .$$

In this case we approximate the partial derivative by a central difference to get

$$v_{0,j} = v_{2,j} - 2\Delta x h(y_j) . \quad (6)$$

Here $v_{0,j}$ is a fictitious point just outside the region. Equation (6) is used to eliminate $v_{0,j}$ in equation (3) when $i = 1$.

In this case the first row of F must be filled with the forcing function f as in equation (4). (This is due to the fact that the equation is assumed to hold at the boundary $x = a$ and we must solve for the solution there.) The array BDA in the parameter list is defined by

$$BDA(J) = h(y_J) .$$

The exact range of the index J is determined by the other boundary conditions, e.g., if the solution is specified on $y = c$, then J starts at 2, whereas, if the normal derivative is specified, then J starts at 1.

Method
(continued)

Periodic Solution: We then assume that

$$u(a+x,y) = u(b+x,y) .$$

In this case we set

$$v_{0,j} = v_{M,j} \quad \text{and} \quad v_{M+2,j} = v_{2,j} . \quad (7)$$

In this case the first row of F is filled as in equation (4). In addition, we must assume that $f(x,y)$ is periodic in x , hence we must have

$$F(1,J) = F(M+1,J) = f(a,y_J)$$

Again, the range of J depends on the other boundary conditions.

If one assumes a combination of normal derivative and periodicity boundary conditions, the linear system of equations is singular. PWSCRT determines the proper constant to be subtracted from the right side in order to ensure that a solution exists for the system. This constant is returned in PERTRB. (The method by which the constant is determined may be found in "The Direct Solution of the Discrete Poisson Equation on a Disk," by P. Swarztrauber and R. Sweet, SIAM J. on Numerical Analysis, (to appear).)

Example

Suppose we wish to approximate the solution of

$$\nabla^2 u - 4u = x^2 \cos \frac{\pi}{2} (y+1)$$

in the rectangle $0 < x < 2$, $-1 < y < 3$, with the boundary conditions

$$u(0,y) = 0$$

$$\frac{\partial u}{\partial x}(2,y) = 3.7$$

and periodicity in the y-direction. If we choose to use a grid with 100 panels in the x-direction (hence, $\Delta x = \frac{2}{100}$) and 80 in the y-direction, (hence $\Delta y = \frac{4}{80}$), then the parameters which would be passed to PWSCRT have the following values:

$$A = 0. \quad , \quad B = 2. \quad , \quad M = 100 \quad , \quad MBDCND = 2 \quad ,$$

BDA undefined since it is not used

$$BDB(J) = 3.7 \quad , \quad J = 1,2,\dots,N+1$$

$$C = -1, \quad D = 3., \quad N = 80, \quad NBDCND = 0,$$

BDC and BDD undefined also

$$ELMBDA = -4$$

$$F(1,J) = 0.$$

$$F(I,J) = [(I-1)\Delta x]^2 \cos\left\{\frac{\pi}{2} [-1 + (J-1)\Delta y + 1]\right\} \quad \left. \vphantom{F(I,J)} \right\} J = 1,2,\dots,81$$

(Note that $F(I,1) = F(I,81)$.)

On output the array F would contain the solution at each point (x_i, y_j) , $i = 1,2,\dots,M+1$; $j = 1,2,\dots,N+1$.

Three Dimensional Equations: PWSCRT may be used to solve Poisson's equation in three dimensions by Fourier transforming in one direction and solving the uncoupled Helmholtz equations using this subroutine.

SUBROUTINE PWSCSP (INTL,TS,TF,M,MBDCND,BDTS,BDTF,RS,RF,N,NBDCND,BDRS,
BDRF,ELMBDA,F,IDIMF,PERTRB,IERROR,W)

DIMENSION OF
ARGUMENTS

BDTS(N+1),BDTF(N+1),BDRS(M+1),BDRF(M+1),F(IDIMF,N+1),
W (see argument list)

LATEST REVISION

March 1974

PURPOSE

PWSCSP determines a solution to a finite difference approximation to Helmholtz's equation in spherical coordinates assuming axi-symmetry (no dependence on longitude).

ACCESS CARDS

*FORTRAN,S=ULIB,N=PWSCSP
*COSY

USAGE

CALL PWSCSP (INTL,TS,TF,M,MBDCND,BDTS,BDTF,RS,RF,N,NBDCND,
BDRS,BDRF,ELMBDA,F,IDIMF,PERTRB,IERROR,W)

ARGUMENTS

On Input

INTL

- = 0 On the first call of PWSCSP.
- = 1 If the arguments TS, TF, M, MBDCND, RS, RF, N, NBDCND are unchanged from a previous call.
- = 2 If the arguments RS, RF, N, NBDCND are unchanged from a previous call.

A call with INTL=0 takes approximately three times as much time as a call with INTL=1.

A call with INTL=2 takes about the same time as a call with INTL=1. Once a call with INTL=0 has been made then subsequent solutions corresponding to different F, BDTS, BDTF, BDRS, BDRF can be obtained faster with INTL≠0 since initialization is not repeated.

TS, TF

The range of THETA (colatitude), i.e., THETA is greater than or equal to TS and less than or equal to TF. TS and TF are in radians. A TS of zero corresponds to the north pole and a TF of PI corresponds to the south pole.

M

The number of panels in the interval [TS, TF]. Hence, there are M+1 grid points at the colatitudes $THETA(I)=(I-1)*DTH+TS$ for $I=1,2,\dots,M+1$. DTH is the width of a panel given by $(TF-TS)/M$.

MBDCND (Refer to notes below)

Indicates the type of boundary condition at THETA=TS and THETA=TF.

- = 1 The solution is specified at TS (see note 1) and the solution is also specified at TF (see note 2).
- = 2 The solution is specified at TS (see note 1) and the derivative of the solution with respect to THETA is specified at TF (see note 4).
- = 3 The derivative of the solution with respect to THETA is specified at TS (see note 3) and the derivative is also specified at TF (see note 4).

- = 4 The derivative of the solution with respect to THETA is specified at TS (see note 3) and the solution is specified at TF (see note 2).
- = 5 The solution is unspecified at TS=0 (see note 5) and the solution is specified at TF (see note 2).
- = 6 The solution is unspecified at TS=0 (see note 5) and the derivative of the solution with respect to THETA is specified at TF (see note 4).
- = 7 The solution is specified at TS (see note 1) and the solution is unspecified at TF=PI (see note 6).
- = 8 The derivative of the solution with respect to THETA is specified at TS (see note 3). The solution is unspecified at TF=PI (see note 6).
- = 9 The solution is unspecified at TS=0 (see note 5) and unspecified at TF=PI (see note 6).

Notes on MBDCND

1. The solution at TS must be stored in $F(1,J)$, $J=1,\dots,N+1$. This boundary condition cannot be used with NBDCND=5 or 6 as the latter imply that the solution is unspecified at $R=0$.
2. The solution at TF must be stored in $F(M+1,J)$, $J=1,\dots,N+1$. This boundary condition cannot be used with NBDCND=5 or 6 as the latter implies that the solution is unspecified at $R=0$.
3. The derivative must be stored in the array $BDTS(J)$ when $J=1,\dots,N+1$ and $F(1,J)$, $J=1,\dots,N+1$ must contain the right side of Helmholtz' equation at $THETA=TS$. Do not use if $TS=0$ but instead, use MBDCND=5, 6 or 9.
4. The derivative must be stored in the array $BDTF(J)$, $J=1,\dots,N+1$ and $F(M+1,J)$, $J=1,\dots,N+1$ must contain the right side of Helmholtz' equation at $THETA=TF$. Do not use if $TF=PI$ but instead, use MBDCND=7, 8 or 9.
5. TS must be zero and $F(1,J)$, $J=1,\dots,N+1$ must contain the right side of Poisson's equation at $THETA=0$.
6. TF must be PI and $F(M+1,J)$, $J=1,\dots,N+1$ must contain the right side of Poisson's equation at $THETA=PI$.

On Input
(continued)

BDTS, BDTF

One dimensional arrays which contain the derivatives of the solution with respect to THETA at TS and TF. If a derivative is not specified, then the corresponding argument(s) may be dummy.

RS, RF

The range of the variable R, i.e., R is greater than or equal to RS and less than or equal to RF.

N

The number of panels in the interval [RS, RF]. Hence there are N+1 grid points at the radii $R(J)=(J-1)*DR+RS$ for $J=1,2,\dots,N+1$. DR is width of a panel given by $(RF-RS)/N$. Let K be an integer greater than one, then N must have the following form depending on NBDCND. If NBDCND=2, 4 or 6, then N must have the form $(2**K)-1$. If NBDCND=1 or 5, then N must have the form $2**K$. If NBDCND=3, then N must have the form $(2**K)-2$.

NBDCND (Refer to notes below)

Indicates the type of boundary condition at radius $R=RS$ and $R=RF$.

- = 1 The solution is specified at RS (see note 1) and also at RF (see note 2).
- = 2 The solution is specified at RS (see note 1) and the derivative of the solution with respect to R is specified at RF (see note 4).
- = 3 The derivative of the solution with respect to R is specified at RS (see note 3) and the derivative is also specified at RF (see note 4).
- = 4 The derivative of the solution with respect to R is specified at $R=RS$ (see note 3) and the solution is specified at RF (see note 2).
- = 5 The solution is unspecified at $R=0$ (see note 5) and the solution is specified at RF (see note 2).
- = 6 The solution is unspecified at $R=0$ (see note 5) and the derivative of the solution with respect to R is specified at $R=RF$ (see note 4).

Notes on NBDCND

1. The solution at RS must be stored in $F(I,1)$, $I=1, \dots, M+1$.
2. The solution at RF must be stored in $F(I,N+1)$, $I=1, \dots, M+1$.
3. The derivative must be stored in the array $BDRS(I)$, $I=1, \dots, M+1$ and $F(I,1)$, $I=1, \dots, M+1$ must contain the right side of Helmholtz' equation at $R=RS$.
4. The derivative must be stored in the array $BDRF(I)$, $I=1, \dots, M+1$ and $F(I,N+1)$, $I=1, \dots, M+1$ must contain the right side of the Helmholtz equation at $R=RF$.
5. RS must be zero and $F(I,1)$, $I=1, \dots, M+1$ must all contain the same value, namely the right side of the equation. Do not use with $MBDCND=1, 2, 4, 5$ or 7 as the latter imply that the solution is specified at $R=0$.

BDRS, BDRF

One dimensional arrays which contain the derivative of the solution with respect to R at $R=RS$ or RF . If a derivative is not specified then the corresponding argument(s) may be dummy.

ELMBDA

The constant in Helmholtz' equation. If $ELMBDA$ is greater than zero then a solution may not exist, however, $PWSCSP$ will attempt to determine it. If $NBDCND=5$ or 6 or $MBDCND=5, 6, 7, 8$ or 9 , then $ELMBDA$ must be zero. (Poisson's equation).

F

A two dimensional array which contains the right side of Helmholtz' equation. $F(I,J)$ corresponds to the value of F at $THETA(I)$ and $R(J)$ (see the definitions of M and N). F must be dimensioned at least $F(M+1,N+1)$.

IDIMF

The row (or first) dimension of the array F as it appears in the program that calls $PWSCSP$. This parameter is used to specify the variable dimension of F . $IDIMF$ must be at least $M+1$.

On Input
(continued)

W

An array which must be provided by the user for work space. Let $L=2**K$ (see the definition of N). Then W must have at least $[2(L+1)(K-1)+3*(N+1)+11*(M+1)+6]$ locations.

On Output

F

Contains the solution $U(I,J)$ at $THETA(I)$ and $R(J)$.

PERTRB

If combinations of unspecified, periodic and derivative boundary conditions are specified for Poisson's equation ($ELMBDA=0$), then a solution may not exist. In this event, PWSCSP computes and subtracts from F a constant PERTRB which ensures that a solution exists. PWSCSP then computes this solution which is a least squares solution to the unperturbed equation.

IERROR

An error flag which indicates invalid input parameters. Except for numbers 0 and 10, a solution is not attempted.

- = 0 No error.
- = 1 TS is less than zero.
- = 2 TS is greater than or equal to TF.
- = 3 MBDCND is less than 1 or greater than 9.
- = 4 RS is less than zero.
- = 5 RS is greater than or equal to RF.
- = 6 NBDCND is less than 1 or greater than 6.
- = 7 N is not of the proper form.
- = 8 An MBDCND of 1, 2, 4, 5 or 7 is used with an NBDCND of 5 or 6.
- = 9 ELMBDA is non-zero and either MBDCND=5, 6, 7, 8 or 9 or NBDCND=5 or 6.
- =10 ELMBDA is greater than zero.

Since IERROR is the only means of detecting an incorrect call to PWSCSP, it should be tested after a call.

W

Contains intermediate values which must not be destroyed if PWSCSP is to be called again with $INTL \neq 0$.

ENTRY POINTS	PWSCSP, FWCSL, BLKTRI, BLKTRI, PROD, PRODP, CPROD, CPRODP, COMPB, IDX, PADD, PPADD
COMMON BLOCKS	None
I/O	None
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Paul N. Swarztrauber, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Methods developed at NCAR during Winter, 1972-1973.
ALGORITHM	PWSCSP develops the linear system of equations which is then solved by a generalized cyclic reduction algorithm in the subroutine BLKTRI.
SPACE REQUIRED	$11326_8 = 4822_{10}$

ACCURACY AND TIMING

The running time is proportional to $MN \log N$. The following table was obtained by solving Poisson's equation on the region $0 \leq r \leq 1$; $0 \leq \theta \leq \pi$ with the solution specified at $r = 1$. The times are in milliseconds of 7600 time and were obtained with $INTL = 1$. A solution with $INTL = 0$ would take about three times that shown in the table. The 7600 has between 14 and 15 decimal digits of accuracy.

<i>M</i>	<i>N</i>	<i>Solution Time</i> (msec)	<i>Relative Error</i>
32	32	60	1.63×10^{-13}
64	64	283	7.62×10^{-13}
128	128	1338	7.89×10^{-11}

PORTABILITY

There are no machine dependent constants.

REQUIRED RESIDENT ROUTINES

SIN, COS, ATAN, SQRT

Method

Subroutine PWSCSP determines an approximate solution to Helmholtz' equation

$$\frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial u}{\partial r} \right) + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left(\sin \theta \frac{\partial u}{\partial \theta} \right) + \frac{\lambda}{r^2 \sin^2 \theta} u = f(r, \theta) \quad (1)$$

on a cross-section of the sphere defined by $TS \leq \theta \leq TF$ and $RS \leq r \leq RF$.

Select integers M, N and define net spacings $\Delta\theta=(TF-TS)/M$, $\Delta r=(RF-RS)/N$ and the net:

$$\theta_i = (i-1)\Delta\theta + TS \quad i=1,2,\dots,M+1$$

$$r_j = (j-1)\Delta r + RS \quad j=1,2,\dots,N+1$$

PWSCSP determines an approximate solution $u_{i,j} \approx u(\theta_i, r_j)$

which satisfies the following finite difference approximation to (1).

$$\begin{aligned} & \frac{1}{r_j^2 \Delta r^2} \left[(r_j + \frac{\Delta r}{2})^2 (u_{i,j+1} - u_{i,j}) - (r_j - \frac{\Delta r}{2})^2 (u_{i,j} - u_{i,j-1}) \right] \\ & + \frac{1}{r_j^2 \sin^2 \theta_i \Delta \theta^2} \left[\sin(\theta_i + \frac{\Delta \theta}{2}) (u_{i+1,j} - u_{i,j}) \right. \\ & \left. - \sin(\theta_i - \frac{\Delta \theta}{2}) (u_{i,j} - u_{i-1,j}) \right] \\ & + \frac{\lambda}{r_j^2 \sin^2 \theta_i} u_{i,j} = f(\theta_i, r_j) \end{aligned} \quad (2)$$

This equation must be augmented by the equations which result from the boundary conditions. Several examples will be given to illustrate how PWSCSP incorporates boundary conditions. If desired, the following equations can be used to check the solution.

1. Assume $TS > 0$ and the derivative of the solution is specified on the boundary $\theta = TS$.

$$\frac{\partial u}{\partial \theta}(TS, r) = h(r) \quad RS < r < RF \quad (3)$$

Method
(continued)

Then the user must specify $BDTS(J)=h(r_J)$ for $J=1,\dots,N+1$. FWSCSP uses this data in the following manner. A centered difference approximation to the derivative in (3) yields

$$u_{0,j} = u_{2,j} - 2\Delta\theta BDTS(J) \quad (4)$$

where $u_{0,j}$ is a virtual point which lies outside the region. This value is eliminated using (4) and (2) evaluated at $i=1$. In addition to specifying $BDTS(J)$, the user must also specify the right side of Helmholtz' equation on the boundary TS.

$$F(1,J) = f(TS,r_J) \quad J=1,\dots,N+1$$

2. Assume now that $TS=0$, $RS>0$ and the solution is unspecified. In this event, equation (2) cannot be used unless $\lambda=0$. If in addition $\frac{\partial u}{\partial \theta}(0,r)=0$ then L'Hospital's rule can be applied to the second term in (1) to obtain

$$\frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial u}{\partial r} \right) + \frac{2}{r^2} \frac{\partial^2 u}{\partial \theta^2} = f(0,r) \text{ at } \theta=0 \quad (5)$$

The sample problem at the end of this write-up illustrates how in practice, the conditions $\lambda=0$ and $\frac{\partial u}{\partial \theta}(0,r)=0$ are satisfied.

The finite difference form of (5) is

$$\frac{1}{r_j^2 \Delta r} \left[(r_j + \frac{\Delta r}{2})^2 (u_{1,j+1} - u_{1,j}) - (r_j - \frac{\Delta r}{2})^2 (u_{1,j} - u_{1,j-1}) \right] + \frac{4}{r_j^2 \Delta \theta^2} (u_{2,j} - u_{1,j}) = f(0, r_j) \quad (6)$$

where the virtual point $u_{0,j}$ was eliminated by using the usual finite difference approximation of $\frac{\partial u}{\partial \theta}(0, r) = 0$ or $u_{0,j} = u_{2,j}$.

3. Assume now that $RS=0$ so that both (1) and (5) cannot be used. If we multiply (1) by $r \sin \theta$ and integrate over the sector $0 < r < \frac{\Delta r}{2}$ and $TS < \theta < TF$,

$$\frac{\Delta r^2}{4} \int_{TS}^{TF} \sin \theta \frac{\partial u}{\partial r}(\theta, \frac{\Delta r}{2}) d\theta + \int_0^{\frac{\Delta r}{2}} \left[\sin(TF) \frac{\partial u}{\partial \theta}(TF, r) - \sin(TS) \frac{\partial u}{\partial \theta}(TS, r) \right] dr = \int_0^{\frac{\Delta r}{2}} \int_{TS}^{TF} \sin \theta f(\theta, r) dr d\theta \quad (7)$$

Denote the first, second and third integrals in (7) by I_1 , I_2 , and I_3 respectively.

$$I_1 \cong \frac{1}{4} \Delta r \Delta \theta \left[w_1 (u_{1,2} - u_c) + w_2 (u_{M+1,2} - u_c) + \sum_{i=2}^M \sin(\theta_i) (u_{i,2} - u_c) \right]$$

where u_c is the approximate solution at the center of the sphere $r=0$. The weights w_1 and w_2 have the value $\frac{1}{2} \frac{\sin \frac{\Delta \theta}{2}}{\cos \frac{\Delta \theta}{2}}$ if a derivative condition is specified.

Method*(continued)*

If θ is zero or π then w_1 or w_2 have the value $\frac{1}{4}\sin\frac{\Delta\theta}{2}$.

These weights are chosen so that the coefficient matrix of the linear system is self adjoint under an inner product which yields a least squares solution via a constant perturbation of F (in the event that the system is singular). Reference [1] contains a description of how the least squares solution is obtained on the surface of the sphere.

Next, we consider I_2 . If $u(\theta, r)$ is smooth at $r=0$, then the limit as r goes to zero of $\frac{\partial u}{\partial \theta}$ must be zero since

$$\frac{\partial u}{\partial \theta} = r(\cos\theta \frac{\partial u}{\partial x} - \sin\theta \frac{\partial u}{\partial z})$$

Hence, a linear approximation on the interval $0 < r < \Delta r$ has the form

$$\frac{\partial u}{\partial \theta}(TS, r) \cong \frac{BDTS(\Delta r)}{\Delta r} r$$

$$\frac{\partial u}{\partial \theta}(TF, r) \cong \frac{BDTF(\Delta r)}{\Delta r} r$$

Substituting these into I_2 we obtain

$$I_2 \cong \frac{\Delta r}{8} [\sin(TF)BDTF(\Delta r) - \sin(TS)BDTS(\Delta r)]$$

In I_3 we approximate $f(\theta, r)$ by its value f_c at the center of the sphere to obtain

$$I_3 \cong \frac{\Delta r^3}{24} [\cos(TS) - \cos(TF)] f_c$$

Finally, substituting the approximate values for I_1 , I_2 , and I_3 and dividing by $\frac{\Delta r^3}{24} [\cos(TS) - \cos(TF)]$ we obtain the equation which is used at the center of the sphere

$$\begin{aligned} & \frac{6\Delta\theta}{\Delta r^2 [\cos(TS) - \cos(TF)]} \left[w_1(u_{1,2}^{-u_c}) + w_2(u_{M+1,2}^{-u_c}) \right. \\ & \quad \left. + \sum_{i=2}^M \sin(\theta_i)(u_{i,2}^{-u_c}) \right] \\ & \quad + \frac{3}{\Delta r^2 [\cos(TS) - \cos(TF)]} [\sin(TF)BDTF(\Delta r) \\ & \quad - \sin(TS)BDTS(\Delta r)] = f_c \end{aligned}$$

The user, of course, need only specify

$$\begin{aligned} BDTS(J) & \quad J=1, \dots, N+1 \\ BDTF(J) & \quad J=1, \dots, N+1 \\ F(I,1) = f(\theta_I, 0) = f_c & \quad I=1, \dots, M+1 \end{aligned}$$

Singular Systems

When combinations of derivative and unspecified boundary conditions are used, then the resulting system of linear equations is singular. In this case, a solution may not exist unless the right side of Helmholtz' equation is perturbed. PWSCSP will compute a constant PERTRB and subtract it from the right side $f(\theta, r)$. PERTRB is computed so that the solution to the perturbed system is a least squares solution to the unperturbed system. The user

Method
(continued)

can determine when this condition exists by printing PERTRB. Additional details for the related problem on the surface of the sphere are given in reference [1].

Sample Problem

In this section we will describe how to use PWSCSP to solve the 3 dimensional Poisson equation on the interior of the sphere subject to Dirichlet boundary conditions on the surface of the sphere. In addition, we will assume equatorial symmetry. Hence we wish to approximate a function $u(\theta, r, \phi)$ which satisfies

$$\begin{aligned} \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial u}{\partial r} \right) + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left(\sin \theta \frac{\partial u}{\partial \theta} \right) & & r < r_0 \\ & & 0 < \theta < \pi/2 \\ + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 u}{\partial \phi^2} = f(\theta, r, \phi) & & 0 < \phi < 2\pi \end{aligned}$$

subject to the boundary conditions

$$u(\theta, r_0, \phi) = g(\theta, \phi) \quad \text{at } r=r_0$$

$$\frac{\partial u}{\partial \theta}(\pi/2, r, \phi) = 0 \quad \text{equatorial symmetry}$$

Define the grid spacings $\Delta\theta = \pi/(2M)$, $\Delta r = r_0/N$ and $\Delta\phi = 2\pi/L$ and the grid

$$\begin{aligned} \theta_i &= (i-1)\Delta\theta & i=1, \dots, M+1 \\ r_j &= (j-1)\Delta r & j=1, \dots, N+1 \\ \phi_k &= (k-1)\Delta\phi & k=1, \dots, L+1 \end{aligned}$$

Then we require that the approximate solution $u_{i,j,k} \approx u(\theta_i, r_j, \phi_k)$ satisfy the finite difference approximation

$$\begin{aligned}
 & \frac{1}{r_j^2 \Delta r^2} \left[(r_j + \frac{\Delta r}{2})^2 (u_{i,j+1,k} - u_{i,j,k}) \right. \\
 & \quad \left. - (r_j - \frac{\Delta r}{2})^2 (u_{i,j,k} - u_{i,j-1,k}) \right] + \frac{1}{r_j^2 \sin \theta_i \Delta \theta^2} \\
 & \quad \cdot \left[\sin(\theta_i + \frac{\Delta \theta}{2}) (u_{i+1,j,k} - u_{i,j,k}) \right. \\
 & \quad \left. - \sin(\theta_i - \frac{\Delta \theta}{2}) (u_{i,j,k} - u_{i-1,j,k}) \right] \\
 & \quad + \frac{1}{r_j^2 \sin^2 \theta_i \Delta \phi^2} (u_{i,j,k+1} - 2u_{i,j,k} \\
 & \quad + u_{i,j,k-1}) = f_{i,j,k}
 \end{aligned} \tag{8}$$

Method
(continued)

For purposes of exposition we will assume that $f_{i,j,k}$ has the form $f_{i,j} \sin \omega k \Delta \phi$ and $g_{i,k}$ has the form $g_i \sin \omega k \Delta \phi$. The case of general $f_{i,j,k}$ and $g_{i,k}$ will be discussed at the end of this section. Note that $f_{i,j}$ and g_i are just the Fourier coefficients of $f_{i,j,k}$ and $g_{i,k}$. We seek a solution $u_{i,j,k}$ of the form $u_{i,j} \sin \omega k \Delta \phi$. Substituting the forms of $u_{i,j,k}$, $f_{i,j,k}$ and $g_{i,k}$ into (8) and dividing by $\sin \omega k \Delta \phi$, we find that the Fourier coefficients must satisfy

$$\begin{aligned} & \frac{1}{r_j \Delta r^2} \left[(r_j + \frac{\Delta r}{2})^2 (u_{i,j+1} - u_{i,j}) \right. \\ & \quad \left. - (r_j - \frac{\Delta r}{2})^2 (u_{i,j} - u_{i,j-1}) \right] \\ & + \frac{1}{r_j^2 \sin^2 \theta_i \Delta \theta^2} \left[\sin(\theta_i + \frac{\Delta \theta}{2}) (u_{i+1,j} - u_{i,j}) \right. \\ & \quad \left. - \sin(\theta_i - \frac{\Delta \theta}{2}) (u_{i,j} - u_{i-1,j}) \right] \\ & \frac{2(1 - \cos \omega \Delta \phi)}{r_j^2 \sin^2 \theta_i \Delta \phi^2} u_{i,j} = f_{i,j} \end{aligned} \tag{9}$$

and on the boundaries

$$u_{i,N+1} = g_i \quad \text{and}$$

$$u_{M+2,j} = u_{M,j}$$

However, since (9) has the same form as (2), PWSCSP can be used to determine $u_{i,j}$ after which the solution is given by the Fourier synthesis $u_{i,j,k} = u_{i,j} \sin \omega k \Delta \phi$. The calling sequence for PWSCSP for the two cases $\omega=0$ and $\omega>0$ will be described.

In the first case ($\omega=0$ and therefore $\lambda=0$) the boundary condition at $\theta=0$ is unspecified and the derivative is specified at $\theta=\pi/2$. Assume that a solution is desired at 5° intervals, hence $L=72$ and $M=18$. Assume also that $N=16$. The parameters are then

INTL=0	
TS=0	
TF= $\pi/2$	
M=18	
MBDCND=6	
BDTS	dummy variable
BDTF(J)=0	J=2,...,N
RS=0	
RF= r_0	
N=16	
NBDCND=5	
BDRS	dummy variable
BDRF	dummy variable
ELMBDA=0	
$F(I,J)=f(\theta_I, r_J)$	I=1,...,M+1 ; J=1,...,N
$F(I,N+1)=g_I$	I=1,...,M+1
IDIMF=19	at least
PERTRB	returned as zero since problem is non-singular.
IERROR	should be returned as zero.
W	work space which must be provided by the user. It must have at least 368 locations.

Method
(continued)

In the second case, $\omega > 0$, the boundary conditions at $\theta = 0$ must be $u_{i,j} = 0$ for otherwise the solution $u_{i,j,k}$ would be multiply valued at $\theta = 0$. Hence the solution is specified at $TS = 0$. The parameters are:

INTL=0	
TS=0	
TF= $\pi/2$	
M=18	
MBDCND=2	
BDTS	dummy
BDTF(J)=0	J=2,...,N
RS=0	
RF= r_0	
N=16	
NBDCND=1	
BDRS	dummy variable
BDRF	dummy variable
ELMBDA= $-2(1-\cos\omega\Delta\phi)/\Delta\phi^2$	
F(I,1)=0	I=1,...,M+1
F(I,N+1)= g_I	I=1,...,M+1
F(1,J)= $f(\theta_I, r_J)$	I=2,...,M+1 ; J=2,...,N
IDIMF=19	at least
PERTRB	returned as zero since problem is non-singular.
IERROR	should be returned as zero.
W	work space which must be provided by the user. It must have at least 368 locations.

For general $f_{i,j,k}$ one first Fourier transforms in the ϕ direction to obtain the Fourier coefficients $f_{i,j}^{(1)}(n)$ and $f_{i,j}^{(2)}(n)$ in the expression

$$f_{i,j,k} = \sum_{n=0}^{L/2} f_{i,j}^{(1)}(n) \sin(nk\Delta\phi) + f_{i,j}^{(2)}(n) \cos(nk\Delta\phi)$$

Similarly, one obtains the Fourier coefficients $g_i^{(1)}(n)$, $g_i^{(2)}(n)$ in the expression

$$g_{i,k} = \sum_{n=0}^{L/2} g_i^{(1)}(n) \sin(nk\Delta\phi) + g_i^{(2)} \cos(nk\Delta\phi)$$

Then each of the Fourier coefficients $u_{i,j}^{(1)}(n)$ and $u_{i,j}^{(2)}(n)$ in the expression

$$u_{i,j,k} = \sum_{n=0}^{L/2} u_{i,j}^{(1)}(n) \sin(nk\Delta\phi) + u_{i,j}^{(2)}(n) \sin(nk\Delta\phi) \quad (10)$$

satisfies (9) and hence PWSCSP can be called repeatedly (L times) to obtain them. Finally, a Fourier synthesis of (10) yields the solution.

References

1. "The direct solution of the discrete Poisson equation on the surface of a sphere," by Paul N. Swarztrauber, SIAM Journal on Numerical Analysis (to appear).
2. "A direct method for the discrete solution of separable elliptic equations" by Paul N. Swarztrauber, SIAM Journal on Numerical Analysis (to appear).

PWSCYL

**SUBROUTINE PWSCYL (INTL,ICOORD,A,B,M,MBDCND,BDA,BDB,C,D,N,NBDCND,
BDC,BDD,ELMBDA,F,IDIMF,PERTRB,IERROR,W)**

**DIMENSION OF
ARGUMENTS**

BDA(N+1),BDB(N+1),BDC(M+1),BDD(M+1),F(IDIMF,N+1),
W(5.25*N+8*M)

LATEST REVISION

September 1, 1973

PURPOSE

PWSCYL solves the usual five-point finite-difference approximation to Helmholtz's equation.

$$\nabla^2 u + \lambda u = f$$

in polar coordinates (r,θ) or cylindrical coordinates (r,z) .

ACCESS CARDS

*FORTRAN,S=ULIB,N=PWSCYL
*COSY

USAGE

CALL PWSCYL (INTL,ICOORD,A,B,M,MBDCND,BDA,BDB,C,D,N,NBDCND,
BDC,BDD,ELMBDA,F,IDIMF,PERTRB,IERROR,W)

ARGUMENTS

On Input

INTL

= 0 if this call is the initial entry to PWSCYL with given values of N and NBDCND.

= 1 if N and NBDCND are unchanged from previous call to PWSCYL.

On Input*(continued)***ICOORD**

Specifies which coordinate system is assumed.

= 1 if polar (r, θ)

= 2 if cylindrical (r, z)

A,B

The range of the variable r , i.e., $A < r \leq B$. A must be less than B , and A must be non-negative.

M

The number of panels into which the interval $[A, B]$ is to be subdivided. (Hence, there will be $M + 1$ grid points in the r -direction.)

MBDCND

Indicates the boundary conditions which are assumed at the two boundaries $r = A$ and $r = B$.

= 1 if the solution is specified at both boundaries.

= 2 if the solution is specified at $r = A$ and the normal derivative of the solution is specified at $r = B$.

= 3 if the normal derivative is specified at both boundaries (this cannot be used with $A = 0$ and $ICOORD = 1$ or with $A = 0$, $ICOORD = 2$, and $ELMBDA \neq 0$).

= 4 if the normal derivative is specified at $r = A$ and the solution at $r = B$. (This cannot be used with $A = 0$ and $ICOORD = 1$ or with $A = 0$, $ICOORD = 2$, and $ELMBDA \neq 0$).

= 5 if the solution is unspecified at $r = A$ and the solution is specified at $r = B$. (This applies only when $A = 0$ and $NBDCND = 0$ or 3.)

= 6 if the solution is unspecified at $r = A$ and the normal derivative is specified at $r = B$. (This applies only when $A = 0$ and $NBDCND = 0$ or 3).

BDA,BDB

Arrays of dimension $n + 1$ which give the values of the normal derivatives at $r = A$ and $r = B$, respectively, when these boundary conditions are assumed.

(The following arguments, described in terms of the θ -variable, apply to either the θ or z variable depending on ICOORD.)

C,D

The range of the θ -variable, i.e., $C \leq \theta \leq D$. C must be less than D.

N

The number of panels into which the interval $[C,D]$ is to be subdivided. (Hence, there will be $N + 1$ grid points in the θ -direction.) N must be of the form $2^p 3^q 5^r$ where p , q , and r are any non-negative integers and N must be greater than 2.

NBDCND

Indicates the boundary conditions which are assumed at $\theta = C$ and $\theta = D$.

= 0 if the solution is periodic in θ .

= 1 if the solution is specified at both boundaries.

= 2 if the solution is specified at $\theta = C$ and the normal derivative of the solution is specified at $\theta = D$.

= 3 if the normal derivative is specified at both boundaries.

= 4 if the normal derivative is specified at $\theta = C$ and the solution is specified at $\theta = D$.

BDC,BDD

Arrays of dimension $M + 1$ which give values of the normal derivative at $\theta = C$ or $\theta = D$, respectively, when these conditions are assumed.

ELMBDA

The constant λ in the Helmholtz equation. If $\lambda > 0$, a solution to the equation may not exist, but the subroutine will attempt to solve it.

On Input
(continued)

F

A two-dimensional array giving the values of the forcing function f and the values of the solution on the boundaries (when specified). F must be dimensioned at least $(M + 1) \times (N + 1)$.

IDIMF

The row (first) dimension of the array F in the calling program. This parameter is used for specifying the variable dimension of F . (Note that $IDIMF$ must be at least $M + 1$.)

W

An array which must be provided for work space. It must be dimensioned at least $5.25*N+8*M$.

On Output

F

Contains the solution of the finite difference approximation.

PERTRB

If one specifies combinations of periodic and normal boundary conditions for a Poisson equation ($\lambda = 0$), the approximation is a singular system. Hence, a solution may not exist. $PERTRB$ is a constant which is calculated and subtracted from the forcing function to ensure that a solution does exist.

IERROR

An error flag which indicates invalid parameters. Except for #11, a solution is not attempted.

- = 0 no error.
- = 1 $A < 0$
- = 2 $A \geq B$
- = 3 $MBDCND < 1$ or $MBDCND > 6$
- = 4 $C \geq D$
- = 5 $N \neq 2^P 3^Q 5^R$ or $N \leq 2$
- = 6 $NBDCND < 0$ or > 4
- = 7 $A = 0$, $ICCOORD = 1$, $MBDCND = 3$ or 4
- = 8 $A > 0$, $MBDCND \geq 5$

= 9 A = 0, MBDCND \geq 5, NBDCND \neq 0 and NBDCND \neq 3
 = 10 A = 0, ICOORD = $2\lambda \neq$ 0, MBDCND \geq 3.
 = 11 $\lambda >$ 0

Since this is the only means of indicating a possibly incorrect call to PWSCYL, IERROR should be tested after the call.

W

Contains intermediate values which must not be destroyed if PWSCRT will be called a gain with INTL = 1.

ENTRY POINTS	PWSCYL, NCHECK, POIS, POINTIT, POISGN, TRI
COMMON BLOCKS	None
I/O	None
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Roland Sweet, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Standardized September 1, 1973
ALGORITHM	The linear system of equations is solved by cyclic reduction using the subroutine POIS. A complete description of the algorithm may be found in "A Generalized Cyclic Reduction Algorithm," by R. Sweet, SIAM J. on Numerical Analysis (to appear).

SPACE REQUIRED

$7277_8 = 3775_{10}$ locations are required.

ACCURACY

At least 10 digits for most uses. This subroutine produces a very accurate solution. The number of significant digits in the answer decreases as M and N increase, but for any reasonable problem one should expect at least 10 digits.

TIMING

The running time is proportional to $MN \log_2 N$. A 65×65 grid ($M = N = 64$) with $MBDCND = NBDCND = 1$, takes 139 milliseconds on the NCAR CDC 7600. (Since subroutine POIS takes 137 milliseconds, the set-up time in PWSCYL is negligible.)

PORTABILITY

There are two machine dependent constants located at the beginning of POINTIT and TRI.

REQUIRED RESIDENT ROUTINES

COS

Method

Subroutine PWSCYL is used to solve a five-point finite difference approximation to Helmholtz's equation in polar (r, θ) or cylindrical (r, z) coordinates.

Polar Coordinates: Suppose we wish to solve

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2} + \lambda u = f(r, \theta) \quad (1)$$

on the region $a < r < b$, $c < \theta < d$. The constants a , b , c , d and λ are parameters of PWSCYL represented by A, B, C, D, and ELMBDA. The region over which we are solving is annular if $a > 0$, it is a complete disc if $d - c = 2\pi$, and it is a sector of a disc if $d - c < 2\pi$.

We define a grid for our approximation by selecting integers M and N , defining

$$\Delta r = \frac{b-a}{M} \quad , \quad \Delta \theta = \frac{d-c}{N} \quad ,$$

and

$$\left. \begin{aligned} r_i &= a + (i-1)\Delta r \quad , \quad i = 1, 2, \dots, M+1 \\ \theta_j &= c + (j-1)\Delta \theta \quad , \quad j = 1, 2, \dots, N+1 \end{aligned} \right\} \quad (2)$$

The integers M and N appear in the parameter list of PWSCYL.

Case $a > 0$: Let us consider first the case $a > 0$ so that the origin ($r = 0$) is not included in the region. If we use central difference approximations to all derivatives of l and denote the approximation to $u(r_i, \theta_j)$ by v_{ij} , we get the finite difference equation

$$\begin{aligned} \frac{1}{r_i \Delta r^2} \left[\left(r_i - \frac{\Delta r}{2} \right) v_{i-1,j} - 2r_i v_{ij} + \left(r_i + \frac{\Delta r}{2} \right) v_{i+1,j} \right] \\ + \frac{1}{r_i^2 \Delta \theta^2} [v_{i,j-1} - 2v_{ij} + v_{i,j+1}] + \lambda v_{ij} = f(r_i, \theta_j) . \end{aligned} \quad (3)$$

This is the basic equation used to determine the v_{ij} . The array F in the parameter list must be dimensioned at least $(M+1) \times (N+1)$. The entries of F for $I = 2, 3, \dots, M$, $J = 2, 3, \dots, N$ are

$$F(I, J) = f(r_I, \theta_J).$$

Equation (3) must be modified at the boundaries to account for the boundary conditions. The two types of boundary conditions which may be assumed at either of the two r -boundaries are (at $r = a$, for example):

Method
(continued)

1. The solution specified--we then are given a function g , such that

$$u(a,\theta) = g(\theta) , c \leq \theta \leq d.$$

In this case we set

$$v_{i,j} = g(\theta_j) , j = 1,2,\dots,N+1$$

The first row of F would then be defined as

$$F(1,J) = g(\theta_J) , J = 1,2,\dots,N+1.$$

2. The normal derivative of the solution is specified--we then are given a function h such that

$$\frac{\partial u}{\partial r}(a,\theta) = h(\theta). \quad (4)$$

In this case we approximate the derivative in (4) by a central difference to get

$$v_{0,j} = v_{2,j} - 2\Delta r h(\theta_j). \quad (5)$$

$v_{0,j}$ is a fictitious point just outside the region. Equation (5) is used to eliminate $v_{0,j}$ in equation (3) when $i = 1$. In this case the solution is unknown at $r = a$, so we must supply PWSCYL with values of the forcing function f . So we define

$$F(1,J) = f(a,\theta_J)$$

and

$$BDA(J) = h(\theta_J).$$

The range of the index J will depend on the boundary conditions assumed at the other boundaries. For instance, if the solution is specified at $\theta = c$, then J starts at 2, whereas, if the normal derivative is specified at $\theta = c$, then J starts at 1.

For the θ -boundaries one can specify boundary conditions 1. and 2. above and, in addition:

3. The solution is periodic in θ . We then assume that

$$u(r,c+\theta) = u(r,d+\theta) \quad .$$

In this case we set

$$v_{i,0} = v_{i,N} \quad \text{and} \quad v_{i,N+2} = v_{i,2} \quad .$$

We must assume that the forcing function f is also periodic in θ , hence we define the array F by

$$F(I,1) = F(I,N+1) = f(r_I, c) \quad .$$

When we assume periodicity in θ , it is not necessary to have $d - c = 2\pi$. For instance, suppose we are solving for the k^{th} wave number in θ . We can then specify $c = 0$, $d = 2\pi/k$, and that the solution be periodic in θ . This will reduce the computation substantially.

Case $a = 0$: Suppose now that the origin is in the region of interest. In this case all the grid points with first component of $r_i = 0$ are identical. Hence, if one assumes boundary condition 1., then we must have

$$F(I,J) = u(0,0) = \text{constant}, \quad J = 1, 2, \dots, N+1 \quad .$$

Method
(continued)

It is not possible to specify the normal derivative at $r = 0$, thus boundary condition 2. is not allowed when $a = 0$.

It is possible to assume that the solution at the origin is unknown (which is certainly the case if $c = 0$, $d = 2\pi$). In this case the unknown value $v_{1,1}$ is computed from the formula

$$\begin{aligned} \left(\lambda - \frac{4}{\Delta r^2}\right) v_{1,1} + \frac{4}{N\Delta r^2} \left(\frac{1}{2}v_{2,1} + v_{2,2} + \dots + v_{2,N} \right. \\ \left. + \frac{1}{2}v_{2,N+1}\right) = f(0,0). \end{aligned} \tag{6}$$

This formula is obtained from discretizing r multiplied by equation (1) integrated over $0 \leq r \leq \frac{\Delta r}{2}$ and $c \leq \theta \leq d$. (The method of solution may be found in "The Direct Solution of the Discrete Poisson Equation on a Disk," by P. Swarztrauber and R. Sweet, SIAM J. on Numerical Analysis (to appear).) In this case the first row of the array F is

$$F(1,J) = f(0,0) \quad , \quad J = 1,2,\dots,N+1 \quad .$$

When the solution is unspecified at $r = 0$ the only possible boundary conditions on the θ -boundaries are either periodicity or normal derivative specified.

Singular Systems: When one specifies combinations of periodic or normal derivative boundary conditions the resulting system of linear equations is singular. In this case a solution may not exist. The subroutine calculates a constant, PERTRB, which is subtracted from the forcing function f to ensure that a solution does exist. This constant is returned to the user. (The method for determining PERTRB is described in the above reference by Swartztrauber and Sweet.)

Cylindrical Coordinates: Subroutine PSWCYL can also solve Helmholtz' equation in cylindrical coordinates

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{\partial^2 u}{\partial z^2} + \frac{\lambda}{r^2} = f(r,z), \quad (7)$$

in the region $a < r < b$, $c < z < d$. The specification of this equation is by the parameter ICOORD. The specification of the grid is exactly the same as for polar coordinates but with θ replaced by z .

The permissible boundary conditions are the same except for the following:

- If $A = 0$ and the solution is unspecified at the origin, the program assumes that the normal derivative of the solution is zero. This follows from consideration of the three dimensional Poisson equation assuming axi-symmetry.
- If one specifies the normal derivative of the solution at $a = 0$ as

$$\frac{\partial u}{\partial r} (0,z) = h(z),$$

the program automatically assumes $h(z) \equiv 0$. This follows from equation (7) applied at $r = 0$, i.e.,

$$\frac{\partial^2 u}{\partial r^2} \Big|_{r=0} + \left(\frac{1}{r} \frac{\partial u}{\partial r} \right) \Big|_{r=0} = f(0,z).$$

Method
(continued)

The second term will be undefined unless $\frac{\partial u}{\partial r} \Big|_{r=0} \equiv 0$, in which case L'Hospital's rule shows that

$$\left(\frac{1}{r} \frac{\partial u}{\partial r} \right) \Big|_{r=0} = \frac{\partial^2 u}{\partial r^2} \Big|_{r=0} .$$

- If $a = 0$ and $\lambda \neq 0$, then the solution (and not the derivative) must be specified at $r = a$.

Three-Dimensional Problems: PWSCYL may be used to solve a three-dimensional Poisson equation by Fourier transforming in one dimension and solving the uncoupled two-dimensional Helmholtz equations using this subroutine.

Examples

- Suppose we want to solve the equation

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2} - 4u = r \cos \theta ,$$

on $0 < r < 1$, $0 \leq \theta \leq 2\pi$, with the boundary conditions

$u(0,0)$ unspecified

$$\frac{\partial u}{\partial r} (1, \theta) = \theta(2\pi - \theta)$$

u periodic in θ .

Suppose we want to subdivide the r -interval $[0,1]$ into 100 panels (so $\Delta r = 1/100$) and the θ -interval $[0,2\pi]$ into 40 panels (so $\Delta \theta = 2\pi/40$). Then the input parameters to PWSCYL would be:

```

INTL = 0
ICCOORD = 1
A = 0., B = 1.
M = 100, MBDCND = 6
BDA - unspecified
BDB(J) = (J-1)Δθ[2π-(J-1)Δθ], J = 1,2,...,N+1.
C = 0., D = 2π
N = 40, NBDCND = 0
BDC, BDD - unspecified
ELMBDA = -4
F(1,J) = 0.
F(I,J) = (I-1)Arcos[(J-1)Δθ], I = 2,3,...,M+1 } J = 1,2,...,N+1

```

- Suppose we want to solve the equation

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{\partial^2 u}{\partial z^2} - u = rz$$

on $1 < r < 3$, $0 < z < 2$, with the boundary conditions

$$u(1,z) = 0$$

$$u(3,z) = z$$

$$\frac{\partial u}{\partial z}(r,0) = r^2$$

$$\frac{\partial u}{\partial z}(r,2) = 0.$$

If we subdivide the r -interval $[1,3]$ into 52 panels and subdivide the z -interval $[0,2]$ into 64 panels, the quantities for input would be

```

INTL = 0
ICCOORD = 2
A = 1., B = 3.
M = 52, MBDCND = 1
BDA, BDB - unspecified
C = 0., D = 2.
N = 64, NBDCND = 3
BDC(I) = [(I-1)Δr]2 } I = 1,2,...,M+1
BDD(I) = 0.
ELMBDA = -1.
F(1,J) = 0.
F(I,J) = (I-1)Δr(J-1)Δz, I = 2,3,...,M } J = 1,2,...,N+1
F(M+1,J) = (J-1)Δz

```


PWSSSP

**SUBROUTINE PWSSSP (INTL,TS,TF,M,MBDCND,BDTS,BDTF,PS,PF,N,NBDCND,BDPS,
BDPF,ELMBDA,F,IDIMF,PERTRB,IERROR,W)**

**DIMENSION OF
ARGUMENTS**

BDTS(N+1),BDTF(N+1),BDPS(M+1),BDPF(M+1),F(IDIMF,N+1),
W(11*(M+1)+6*N)

LATEST REVISION

March 1974

PURPOSE

PWSSSP determines a solution to a finite difference approximation to Helmholtz's equation in spherical coordinates and on the surface of the sphere.

ACCESS CARDS

*FORTRAN,S=ULIB,N=PWSSSP
*COSY

USAGE

CALL PWSSSP (INTL,TS,TF,M,MBDCND,BDTS,BDTF,PS,PF,N,NBDCND,
BDPS,BDPF,ELMBDA,F,IDIMF,PERTRB,IERROR,W)

ARGUMENTS

On Input

INIL

Set INIL=0 on the first call of PWSSSP. Then set INIL=1 on subsequent calls where TS, TF, M, MBDCND, PS, PF, N, NBDCND remain unchanged. The subsequent solutions, corresponding to different right sides F, will be obtained about 2% faster since initialization is not repeated.

TS, TF

The range of THETA (colatitude), i.e., THETA is greater than or equal to TS and less than or equal to TF. TF and TS are in radians. A TS of zero corresponds to the north pole and a TF of PI corresponds to the south pole.

M

The number of panels in the interval [TS, TF]. Hence, there are M+1 grid points at the colatitudes $THETA(I)=(I-1)*DTH+TS$ for $I=1,2,\dots,M+1$. DTH is the width of a panel given by $(TF-TS)/M$.

MBDCND (Refer to notes below)

Indicates the type of boundary condition at THETA=TS and THETA=TF.

- = 1 The solution is specified at TS (see note 1) and the solution is also specified at TF (see note 2).
- = 2 The solution is specified at TS (see note 1) and the derivative of the solution with respect to THETA is specified at TF (see note 4).
- = 3 The derivative of the solution with respect to THETA is specified at TS (see note 3) and the derivative is also specified at TF (see note 4).
- = 4 The derivative of the solution with respect to THETA is specified at TS (see note 3) and the solution is specified at TF (see note 2).
- = 5 The solution is unspecified at TS=0 (see notes 5, 7) and the derivative of the solution is specified at TF (see note 2).

- = 6 The solution is unspecified at $TS=0$ (see notes 5, 7) and the derivative of the solution with respect to θ is specified at TF (see note 4).
- = 7 The solution is specified at TS (see note 1) and the solution is unspecified at $TF=PI$ (see notes 6, 7).
- = 8 The derivative of the solution with respect to θ is specified at TS (see note 3) and the solution is unspecified at $TF=PI$ (see notes 6, 7).
- = 9 The solution is unspecified at $TS=0$ (see notes 5, 7) and unspecified at $TF=PI$ (see notes 6, 7).

Notes on MBDCND

1. The solution at TS must be stored in $F(1,J), J=1, \dots, N+1$.
2. The solution at TF must be stored in $F(M+1,J); J=1, \dots, N+1$.
3. The derivative must be stored in the array $BDTS$ and $F(1,J), J=1, \dots, N+1$ must contain the right side of Helmholtz equation at $\theta=TS$. Do not use if $TS=0$.
4. The derivative must be stored in the array $BDTF$ and $F(M+1,J), J=1, \dots, N+1$ must contain the right side of Helmholtz equation at $\theta=TF$. Do not use if $TF=PI$.
5. TS must be zero and $F(1,J), J=1, \dots, N+1$ must all contain the same value, namely the right side of Helmholtz equation at the north pole.
6. TF must be PI and $F(M+1,J), J=1, \dots, N+1$ must all contain the same value, namely the right side of Helmholtz equation at the south pole.
7. $MBDCND=5,6,7,8$ or 9 cannot be used with $NBDCND=1,2$ or 4 as the former indicates that the solution is unspecified at the pole and the latter indicates that the solution is specified at the pole.

On Input
(continued)

BDTS, BDTF

One dimensional arrays which contain the derivatives of the solution with respect to THETA at TS and TF. If a derivative is not specified then the corresponding argument(s) may be dummy.

PS, PF

The range of PHI (longitude), i.e., PHI is greater than or equal to PS and less than or equal to PF. PS and PF are in radians. If PS=0, and PF=2*PI, then periodic boundary conditions are usually prescribed.

N

The number of panels in the interval [PS, PF]. Hence, there are N+1 grid points at the longitudes defined by $\text{PHI}(J) = (J-1) * \text{DPHI} + \text{PS}$ for $J=1, 2, \dots, N+1$. DPHI is the width of a panel given by $(\text{PF} - \text{PS}) / N$. N must be of the form $2^p 3^q 5^r$ where p, q, and r are non-negative integers and N is greater than 2.

NBDCND (Refer to notes below)

Indicates the type of boundary condition at longitude PHI=PS and PHI=PF.

- = 0 Periodic, i.e., $U(I,1) = U(I,N+1)$.
- = 1 The solution is specified at PS (see note 1) and also at PF (see note 2).
- = 2 The solution is specified at PS (see note 1) and the derivative of the solution with respect to PHI is specified at PF (see note 4).
- = 3 The derivative of the solution with respect to PHI is specified at PS (see note 3) and the derivative is also specified at PF (see note 4).
- = 4 The derivative of the solution with respect to PHI is specified at PS (see note 3) and the solution is specified at PF (see note 2).

Notes on NBDCND

1. The solution at PS is stored in F(I,1) for $I=1, \dots, M+1$.
2. The solution at PF is stored in F(I,N+1) for $I=1, \dots, M+1$.
3. The derivative is stored in the array BDPS and F(I,1), $I=1, 2, \dots, M+1$ must contain the right side of Helmholtz equation at PHI=PS.

4. The derivative is stored in the array BDPF and $F(I,N+1), I=1,2,\dots,M+1$ must contain the right side of Helmholtz equation at $\text{PHI}=\text{PF}$.

BDPS, BDPF

One dimensional arrays which contain the derivative of the solution with respect to longitude at PS and PF. If a derivative is not specified then the corresponding argument(s) may be dummy.

ELMBDA

The constant in Helmholtz equation. If ELMBDA is greater than zero, a solution may not exist, however, PWSSSP will attempt to determine it.

F

A two dimensional array which contains the right side of Helmholtz equation. $F(I,J)$ corresponds to the value of F at $\text{THETA}(I)$ and $\text{PHI}(J)$. (See the definition of M and N.) F must be dimensioned at least $F(M+1,N+1)$.

IDIMF

The row (or first) dimension of the array F as it appears in the program that calls PWSSSP. This parameter is used for specifying the variable dimension of F. IDIMF must be at least M+1.

W

An array which must be provided by the user for work space. It must be dimensioned at least $11*(M+1)+6*N$.

On Output

F

Contains the solution $U(I,J)$ at $\text{THETA}(I)$ and $\text{PHI}(J)$.

PERTRB

If combinations of unspecified, periodic and derivative boundary conditions are specified for Poisson's equation ($\text{ELMBDA}=0$), then a solution may not exist. In this event, PWSSSP computes

On Output
(continued)

and subtracts from F a constant PERTRB which ensures that a solution exists. PWSSSP then computes this solution which is a least squares solution to the unperturbed equation.

IERROR

An error flag which indicates invalid input parameters. Except for number 8, a solution is not attempted.

- = 0 No error.
- = 1 TS is less than zero.
- = 2 TS is greater than or equal to TF.
- = 3 MBDCND is less than 1 or greater than 9.
- = 4 PS is greater than or equal to PF.
- = 5 N is not of the form $2**P*3**Q*5**R$ or $N < 2$.
- = 6 NBDCND is less than zero or greater than 4.
- = 7 An MBDCND of either 5, 6, 7, 8, 9 is used with an NBDCND of either 1, 2, or 4.
- = 8 ELMBDA is greater than zero.

Since IERROR is the only means of detecting an incorrect call to PWSSSP, it should be tested after a call.

W

Contains intermediate values which must not be destroyed if PWSSSP is to be called again with INTL=1.

ENTRY POINTS

PWSSSP, PWSSS1, POIS, POINTIT, TRI, POISGN, NCHECK

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

SPECIALIST

Paul N. Swarztrauber, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Methods were developed at NCAR during Winter, 1972-1973.

ALGORITHM

The reference given at the end of this write-up describes the development of the linear systems which are then solved by cyclic reduction in POIS.

SPACE REQUIRED

 $7344_8 = 3808_{10}$

ACCURACY AND TIMING

The following computational results were obtained by solving Poisson's equation on the entire surface of the sphere on the 7600. The operation count is proportional to $MN \log N$. The 7600 has 14 decimal digits of accuracy and the error is relative. Below is a table containing the computational results.

<i>M</i>	<i>N</i>	<i>Solution Time</i> (msec)	<i>Relative Error</i>
16	32	25	2.98×10^{-13}
32	64	103	9.66×10^{-13}
64	128	446	4.05×10^{-11}

PORTABILITY

There are two machine dependent constants located at the beginning of POINT and TRI.

REQUIRED RESIDENT ROUTINES

SIN, COS, ATAN

Method

Subroutine PWSSSP determines an approximate solution to Helmholtz equation

$$\frac{1}{\sin\theta} \frac{\partial}{\partial\theta} \left(\sin\theta \frac{\partial u}{\partial\theta} \right) + \frac{1}{\sin^2\theta} \frac{\partial^2 u}{\partial\phi^2} + \lambda u = f(\theta, \phi) \quad (1)$$

on all or a portion of the surface of the sphere defined by $TS < \theta < TF$; $PS < \phi < PF$. Select integers M, N and define net spacings $\Delta\theta = (TF - TS)/M$; $\Delta\phi = (PF - PS)/N$ and the net:

$$\begin{aligned} \theta_i &= (i-1)\Delta\theta + TS & i=1,2,\dots,M+1 \\ \phi_j &= (j-1)\Delta\phi + PS & j=1,2,\dots,N+1 \end{aligned} \quad (2)$$

PWSSSP determines an approximate solution $u_{i,j} \cong u(\theta_i, \phi_j)$ which satisfies the following finite difference approximation to (1).

$$\begin{aligned} &\frac{1}{\sin\theta_i \Delta\theta^2} \left[\sin\left(\theta_i + \frac{\Delta\theta}{2}\right) (u_{i+1,j} - u_{i,j}) \right. \\ &\quad \left. - \sin\left(\theta_i - \frac{\Delta\theta}{2}\right) (u_{i,j} - u_{i-1,j}) \right] \\ &\quad + \frac{1}{(\sin\theta_i \Delta\phi)^2} (u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) + \lambda u_{i,j} = f(\theta_i, \phi_j) \end{aligned} \quad (3)$$

This equation must be augmented by equations which result from the boundary conditions. Several examples will be given to illustrate how PWSSSP incorporates the boundary condition. If desired, the following equations can be used to check the solution.

1. Assume $TS > 0$ and the solution is specified on the boundary $\theta = TS$.

$$u(PS, \phi) = g(\phi) \quad PS \leq \phi \leq PF \quad (4)$$

When the solution is specified it is stored in the appropriate locations in F. Hence, we set

$$F(1, J) = g(\phi_J) \quad J = 1, 2, \dots, N+1 \quad (5)$$

2. Assume $TS > 0$ and the derivative of the solution is specified on the boundary $\theta = TS$.

$$\frac{\partial u}{\partial \theta}(TS, \phi) = h(\phi) \quad PS \leq \phi \leq PF \quad (6)$$

A centered difference approximation to (6) yields:

$$u_{0,j} = u_{2,j} - 2\Delta\theta h(\phi_j) \quad (7)$$

where $u_{0,j}$ is a "virtual" point outside the region which is eliminated between (7) and equation (3) evaluated at $i=1$.

The derivative of the solution must be stored in the array BDTS.

$$BDTS(J) = h(\phi_J) \quad J = 1, \dots, N+1 \quad (8)$$

Method
(continued)

In addition, when the derivative of the solution is specified, the right side of Helmholtz equation must be specified on the boundary

$$F(1,J) = f(TS,\phi_J) \quad J = 1,\dots,N+1 \quad (9)$$

3. Assume the solution is periodic in ϕ , i.e.,

$$u(\theta,\phi) = u(\theta,\phi+PF) \quad (10)$$

Then the approximate solution satisfies

$$u_{i,j} = u_{i,j+N} \quad \text{for all } i \text{ and } j+N \text{ modulo } N \quad (11)$$

$F(I,J)$ must be defined for $J=1,\dots,N+1$ and be periodic, i.e., $F(I,1)=F(I,N+1)$.

When periodicity is assumed, it is not necessary to have $PF-PS=2\pi$. However, it is likely that $k(PF-PS)=2\pi$ for some integer k which would correspond to the wave number under consideration. This permits the user to determine solutions for high wave numbers with less computation.

4. Assume now that $TS=0$ and hence (3) cannot be used. Instead, we multiply (1) by $\sin\theta$ and then integrate over the spherical segment $0 < \theta < \Delta\theta/2$; $PS < \phi < PF$ to obtain:

$$\sin\frac{\Delta\theta}{2} \int_{PS}^{PF} \frac{\partial u}{\partial \theta}(\frac{\Delta\theta}{2}, \phi) d\phi + \int_0^{\frac{\Delta\theta}{2}} \frac{1}{\sin\theta} \left[\frac{\partial u}{\partial \phi}(\theta, PF) - \frac{\partial u}{\partial \phi}(\theta, PS) \right] d\theta \quad (12)$$

$$+ \lambda \int_{PS}^{PF} \int_0^{\frac{\Delta\theta}{2}} \sin\theta u(\theta, \phi) d\theta d\phi = \int_{PS}^{PF} \int_0^{\frac{\Delta\theta}{2}} \sin\theta f(\theta, \phi) d\theta d\phi$$

or $I_1 + I_2 + I_3 = I_4$

First we approximate

$$\frac{\partial u}{\partial \theta}(\frac{\Delta\theta}{2}, \phi) \cong \frac{u(\Delta\theta, \phi) - u(0, 0)}{\Delta\theta} \text{ and } \sin\frac{\Delta\theta}{2} \cong \frac{\Delta\theta}{2} \quad (13)$$

and apply the trapezoidal rule to obtain

$$I_1 = \frac{\Delta\phi}{2} \left[\sum_{J=2}^N u(\Delta\theta, \phi_J) + \frac{1}{2}u(\Delta\theta, PS) + \frac{1}{2}u(\Delta\theta, PF) - Nu(0, 0) \right] \quad (14)$$

Now if $u(\theta, \phi)$ is smooth at the poles then

$\lim_{\theta \rightarrow 0} \frac{\partial u}{\partial \phi} = 0$ since $u(\theta, \phi)$ as a function of ϕ approaches a constant value, namely $u(0, \phi)$ at the pole. Hence, a linear approximation on the interval $0 < \theta < \Delta\theta$ has the form:

$$\frac{\partial u}{\partial \phi}(\theta, PS) \cong \frac{BDPS(\Delta\theta)}{\Delta\theta} \theta \quad (15)$$

$$\frac{\partial u}{\partial \phi}(\theta, PS) \cong \frac{BDPF(\Delta\theta)}{\Delta\theta} \theta$$

If, in addition, we assume $\sin\theta \cong \theta$ then

$$I_2 \cong \frac{1}{2} \left[BDPF(\Delta\theta) - BDPS(\Delta\theta) \right] \quad (16)$$

If we approximate $u(\theta, \phi)$ by $u(0, 0)$ then we obtain

$$I_3 = \frac{N\Delta\phi\Delta\theta^2}{8} \lambda u(0, 0) \quad (17)$$

and similarly if we approximate $f(\theta, \phi)$ by $f(0, 0)$ then

$$I_4 = \frac{N\Delta\phi\Delta\theta^2}{8} f(0, 0) \quad (18)$$

Method
(continued)

Finally, we substitute into (12) the approximate values of the integrals and multiply through by $8/(N \cdot \Delta\theta^2 \Delta\phi)$ to obtain the formula which is used at the pole by PWSSSP.

$$\begin{aligned} & \frac{4}{N \cdot \Delta\theta^2} \left[\frac{1}{2}u(\Delta\theta, PS) + \frac{1}{2}u(\Delta\theta, PF) + \sum_{J=2}^N u(\Delta\theta, \phi_J) - N \cdot u(0,0) \right] \\ & + \frac{4}{N \Delta\phi \Delta\theta^2} \left[BDPF(\Delta\theta) - BDPS(\Delta\theta) \right] + \lambda u(0,0) = f(0,0) \end{aligned} \quad (19)$$

A similar formula can be derived at the south pole. The user, of course, need only specify:

$$\begin{array}{ll} BDPS(I) & I=1, \dots, M+1 \\ BDPF(I) & I=1, \dots, M+1 \\ F(1, J) = f(0,0) & J=1, \dots, N+1 \end{array}$$

Singular Systems

When combinations of periodic, derivative, and unspecified boundary conditions are used, then the resulting system of linear equations is singular. In this case, a solution may not exist unless the right side of Helmholtz equation is perturbed. PWSSSP will compute a constant PERTRB and subtract it from the right side. PERTRB is computed so that the solution to the perturbed system is a least squares solution to the unperturbed system. The user can determine when this condition exists by printing PERTRB. Additional details are given in reference [1].

Examples

1. Suppose we wish to solve Poisson's equation on the entire surface of the sphere using a 5° grid. Then $M=36$ and $N=72$. The user must compute the right side of Poisson's equations and store it in $F(I,J)=f(\theta_I, \phi_J)$ where

$$\theta_I = (I-1)\frac{\pi}{M} \quad I=1, \dots, 37$$

$$\phi_J = (J-1)\frac{2\pi}{N} \quad J=1, \dots, 73$$

Note that for $i=1$, $F(1,J)$ is the same for all J and similarly at $I=M+1$. PWSSSP is then called with the following arguments:

INTL = 0	
TS = 0	
TF = π	
M = 36	
MBDCND = 9	includes both poles
BDTS, BDTF	dummy variables
PS = 0	
PF = 2π	
N = 72	
NBDCND = 0	periodic
BDPS, BDPF	dummy variables
ELMBDA = 0	
$F(I,J) = f(\theta_I, \phi_J)$	$I=1, \dots, 37$; $J=1, \dots, 73$
IDIMF = 37	if the first dimension of F is 37 in user's program.
PERTRB	will be returned as the perturbation of F .
IERROR	should be returned as zero.
W	should be dimensioned at least 839 for work space by PWSSSP.

Method
(continued)

2. Assume we wish to solve Poisson's equation on the northern hemisphere only, assuming equatorial symmetry.

Then $\frac{\partial u}{\partial \theta} = 0$ at $\theta = \pi/2$ hence the user must set

$$\text{BDTF}(J) = 0 \quad J = 1, 2, \dots, 73$$

In addition, we must compute

$$F(I, J) = f(\theta_I, \phi_J) \quad I = 1, \dots, 19 \quad ; \quad J = 1, 2, \dots, 73$$

PWSSSP is then called with the following arguments:

INTL = 0	
TS = 0	
TF = $\pi/2$	
M = 18	
MBDCND = 6	
BDTS	dummy variable
BDTF(J) = 0	J=1, ..., 73
PS = 0	
PF = 2π	
N = 72	
NBDCND = 0	periodic
BDPS, BDPF	dummy variables
ELMBDA = 0	
F(I, J) = f(θ_I, ϕ_J)	I=1, 2, ..., 19; J=1, ..., 73
IDIMF = 37	
PERTRB	will be returned as the perturbation of F.
IERROR	should be returned as zero.
W	should be dimensioned at least 641 for work space by PWSSSP.

Reference

"The direct solution of the discrete Poisson equation on the surface of the sphere" by Paul N. Swarztrauber, SIAM Journal on Numerical Analysis (to appear)

7

SPECIAL FUNCTIONS

BESLIK

BESLJY

CXERFC

ELIPE

ELIPK

EXPINT

HYPER

SUBROUTINE BESLIK (X,V,N,A,LDIMA,ERR)**DIMENSION OF ARGUMENTS**

The double precision array A must be dimensioned at least $\text{MAX}(X,N)+16$.

LATEST REVISION

June 6, 1974

PURPOSE

BESLIK computes the modified BESSEL function of the first kind (I_V) and the modified Bessel function of the second kind (K_V) for real argument and real orders. More precisely, it computes a double-precision sequence of exponentially scaled modified Bessel functions; either

$$\{e^{-X}I_V(X), e^{-X}I_{V+1}(X), \dots, e^{-X}I_{V+N}(X)\}$$

or

$$\{e^X K_V(X), e^X K_{V+1}(X), \dots, e^X K_{V+N}(X)\}$$

for $0 \leq V < 1$. and $X > 0$.

ACCESS CARDS

*FORTRAN,S=ULIB,N=BESLK
*COSY

ARGUMENTS

On Input

X

Is the required argument. It must be double-precision and must satisfy $X > 0$.

V

Is the initial order. It must be double-precision and must satisfy $0 \leq V < 1.0$.

N

Is an integer satisfying $N \geq 0$. The subroutine calculates functional values with argument X for the orders V, V+1, V+2, ..., V+N

LDIMA

Specifies the dimension of the double-precision array A (in which the functional values are returned) and must be at least $\text{MAX}(X,N)+16$ to supply additional working space.

ERR

Serves as an error message flag. If set to 1.0 in the functional call, the error printout is suppressed when ERR is 0.0 (or any value other than 1.0), an error message is produced by ULIBER in the event that X, V, N or LDIMA is out of range.

On Output

A

The array A will hold the N+1 functional values for argument X and orders V, V+1, V+2, ..., V+N in A(1), A(2), A(3), ..., A(N+1) respectively. The I values are multiplied by $\text{EXP}(-X)$, the K values are multiplied by $\text{EXP}(X)$.

ERR

ERR has a value -1.0 upon return in the event that calling parameters were out of range (thus precluding the calculation of values).

ENTRY POINTS

BESLJ, BESLY

SPECIAL CONDITIONS

It is important to note that the argument X, the initial order V, and the array A must be declared double-precision in the calling program. Use of the on-line functions DBLE and SNGL for single-double conversions may be convenient. Thus,

$$X = \text{DBLE}(\text{ARG})$$

converts the single-precision value of ARG to the corresponding double-precision value in X, while

$$\text{RESULT} = \text{SNGL}(A(1))$$

converts the double-precision value of A(1) to the (truncated) corresponding single-precision value in RESULT. The computed values of this subroutine will have full single-precision accuracy upon conversion to single-precision.

COMMON BLOCKS

None

I/O

If ERR is set to 1.0 before the call, no printout is produced. With ERR set to 0.0, BESLIK produces an error message in the event that X, V, N or LDIMA is out of range.

PRECISION

Double

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Ned Read, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized June 6, 1974

ALGORITHM

The method used is the recursion technique of J.C.P. Miller and normalization with the addition formula for computing the I's. The value of K_V is computed by quadrature or expansion in a series of I_{V+N} , and K_{V+1} is calculated by the Wronskian. The computer program is based upon one in use at the Argonne National Laboratory, written by Donald F. Jordan. Reference: M. Goldstein and R. M. Thaler, "Recurrence Techniques for the Calculation of BESSEL functions," *MTAC, Vol. 13*, 1959, pp 102-108.

SPACE REQUIRED

$\sim 2574_8 = (1404_{10})$ locations

ACCURACY

Approximately 15 significant figures.

TIMING

For the calculation of the 550 values for $X=1.,2.,\dots,50.$; $V=0,1,\dots,10$. BESLI required 247 milliseconds while BESLK required 325 milliseconds (on the CDC 7600).

PORTABILITY

No machine dependent constants.

REQUIRED RESIDENT ROUTINES

DLOG, DEXP, DSIN

SUBROUTINE BESLJY (X,V,N,A,LDIMA,ERR)**DIMENSION OF ARGUMENTS**

The double-precision array A must be dimensioned at least $\text{MAX}(X,N)+16$.

LATEST REVISION

June 6, 1974

PURPOSE

BESLJY computes the BESSEL functions of the first kind (J_V) and the second kind (Y_V) for real argument and real orders. More precisely, it computes a double-precision sequence

$$\{J_V(X), J_{V+1}(X), \dots, J_{V+N}(X)\}$$

or

$$\{Y_V(X), Y_{V+1}(X), \dots, Y_{V+N}(X)\}$$

for $0 \leq V < 1$ and $X > 0$.

ACCESS CARDS

*FORTRAN,S=ULIB,N=BESLJY
*COSY

USAGE

BESLJY has entry points BESLJ and BESLY.

CALL BESLJ (X,V,N,A,LDIMA,ERR)

CALL BESLY (X,V,N,A,LDIMA,ERR)

ARGUMENTS

On Input

X

Is the required argument. It must be double-precision and must satisfy $X > 0$.

Y

Is the initial order. It must be double-precision and must satisfy $0 \leq V < 1.0$.

N

Is an integer satisfying $N \geq 0$. The subroutine calculates functional values with argument X for the orders V, V+1, V+2, ..., V+N.

LDIMA

Specifies the dimension of the double-precision array A (in which the functional values are returned) and must be at least $\text{MAX}(X,N)+16$ to supply additional working space.

ERR

Serves as an error message flag. If set to 1.0 in the function call, the error printout is suppressed. When ERR is 0.0 (or any value other than 1.0), an error message is produced by ULIBER in the event that X, V, N or LIDMA is out of range.

On Output

A

The double-precision array A will hold the N+1 functional values for argument X and orders V, V+1, V+2, ..., V+N in A(1), A(2), A(3), ..., A(N+1) respectively.

ERR

ERR has a value -1.0 upon return in the event that calling parameters were out of range (thus precluding the calculation of values).

ENTRY POINTS

BESLJ, BESLY

SPECIAL CONDITIONS

Overflow will occur if excessively large values of N are used in calling for the computation of Y_{V+N} , $N=0,1,2,\dots,N$.

It is important to note that the argument X, the initial order V, and the array A must be declared double-precision in the calling program. Use of the on-line functions DBLE and SNGL for single-double conversions may be convenient. Thus,

$$X=\text{DBLE}(\text{ARG})$$

converts the single-precision value of ARG to the corresponding double-precision value in X, while

$$\text{RESULT}=\text{SNGL}(\text{A}(1))$$

converts the double-precision value of A(1) to the (truncated) corresponding single-precision value in RESULT. The computed values of this subroutine will have full single-precision accuracy upon conversion to single-precision.

COMMON BLOCKS

None

I/O

If ERR is set to 1.0 before the call, no printout is produced. With ERR set to 0.0, BESLJY produces an error message in the event that X, V, N or LDIMA is out of range.

PRECISION

Double

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Ned Read, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized June 6, 1974

ALGORITHM

The method used is the recursion technique of J.C.P. Miller and normalization with addition formulae. The compiler program is based upon one in use at the Argonne National Laboratory, written by Donald F. Jordan. Reference: M. Goldstein and R. M. Thaler, "Recurrence Techniques for the Calculation of BESSEL Functions," *MTAC, Vol. 13*, 1959, pp 102-108.

SPACE REQUIRED

~ $2642_8 = (1442_{10})$ locations

ACCURACY

BESLJ has approximately 20 significant figures.
BESLY has approximately 15 significant figures.

TIMING

For the calculation of the 550 values for $X = 1., 2., \dots, 50.;$
 $V=0, 1, \dots, 10.$ BESLJ required 252 milliseconds while BESLY
required 285 milliseconds on the CDC 7600.

PORTABILITY

No machine dependent constants.

**REQUIRED RESIDENT
ROUTINES**

DCOS, DSIN, DLOG

SUBROUTINE CXERFC (ZA,ZB,ERCA,ERCB,ERR)

DIMENSION OF ARGUMENTS None

LATEST REVISION June 24, 1974

PURPOSE CXERFC evaluates the complementary (complex) error function $\text{erfc}(z)$, where

$$\text{erfc}(z) = 1 - \text{erf}(z)$$

$$\begin{aligned} &= 1 - \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \\ &= \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-t^2} dt \end{aligned}$$

Let the argument be $z = x + iy$. If $|z| > 0.5$ and $|x| < |y|$, the subroutine computes $e^{z^2} \text{erfc}(z)$ instead of $\text{erfc}(z)$.

ACCESS CARDS

*FORTRAN,S=ULIB,N=CXERFC
*COSY

USAGE

CALL CXERFC (ZA,ZB,ERCA,ERCB)

ARGUMENTS

On Input

ZA,ZB

Are respectively the real and imaginary parts of the argument. They must be declared double-precision in the calling program.

ERR

Serves as an error message flag. When set to 1.0 in the call, error message printout is suppressed. Otherwise, use ERR=0.0.

On Output

ERCA,ERCB

Are respectively the real and imaginary parts of the functional value, $\operatorname{erfc}(z)$ (or $e^{z^2}\operatorname{erfc}(z)$). They must be declared double-precision in the calling program.

ERR

Is set to -1.0 in the event that convergence to the sought after functional value did not occur.

ENTRY POINTS

CXERFC

SPECIAL CONDITIONS

Let the argument be $z = x + iy$. If $|z| > 0.5$ and $|x| < |y|$, the subroutine computes $e^{z^2}\operatorname{erfc}(z)$ instead of $\operatorname{erfc}(z)$.

COMMON BLOCKS

None

I/O ULIBER is called to produce an error message in the event that convergence is not attained.

PRECISION Double

REQUIRED ULIB ROUTINES ULIBER

SPECIALIST Ned Read, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Originally written by Garney Hardy, of the Environmental Science Services Administration, Boulder, Colorado. Use is made of a subroutine written by Kenneth B. Larson of Massachusetts Institute of Technology.

ALGORITHM The method of computation is dependent on $|z|$ and on $|\arg(z)|$.

SPACE REQUIRED Approximately $8000_8 = (4096_{10})$ locations

ACCURACY Results have been checked against tables for real values of z and against results from independent subroutines. Agreement was found up to ± 1 in the 15^{th} significant figure.

TIMING

Computation of the 200 values corresponding to the real arguments $X = .01, .02, \dots, 2.00$ required 195 milliseconds on the CDC 7600. With $z = x + iy$, computation of the 200 values corresponding to $x = .1, .2, \dots, 2.0$; $y = .1, .2, \dots, 1.0$ required 632 milliseconds on the CDC 7600.

PORTABILITY

No machine dependent constants

**REQUIRED RESIDENT
ROUTINES**

DSQRT, DEXP, DCOS, DSIN

**REQUIRED ULIB
ROUTINES**

ULIBER

LATEST REVISION

September 1972

PURPOSE

This package computes single precision values of the complete elliptic integrals of the second kind:

$$E(k^2) = \int_0^{\frac{1}{2}\pi} [1 - k^2 \sin^2 x]^{1/2} dx \text{ for } 0 \leq k \leq 1$$

ACCESS CARDS

*FORTRAN,S=ULIB,N=ELIPE
*COSY

*FORTRAN,S=ULIB,N=MONERR
*COSY

USAGE

For maximum accuracy, E may be computed as E(CAYSQ), E(CAY**2), or E(1-ETA) by invoking the FUNCTION subprograms:

Y = DELIPE(CAYSQ) when $0 \leq \text{CAYSQ} \leq 1$,
Y = DELIEL(CAY) when $1 \leq \text{CAY} \leq 1$,
Y = DELIEM(ETA) when $0 \leq \text{ETA} \leq 1$,

respectively.

ARGUMENTS

Only the self-explanatory ones noted above.

ENTRY POINTS

NATSEE, DELIPE, DELIE1, DELIEM, MONERR, FCNMON

SPECIAL CONDITIONS

The following table indicates the types of error that may be encountered in this packet and the function value supplied in each case (XIND is -1. -- an impossible function value).

<i>Error No.</i>	<i>Error</i>	<i>Function Values For</i>	
		<i>DELIPE or</i>	<i>DELIE1</i>
		<i>DELIEM</i>	
1	K**2 .LT. 0, or ETA .GT. 1	XIND	----
2	K**2 .GT. 1, or ETA .LT. 0	XIND	XIND

All error processing is handled through FCNMON. The default error procedure is to print an error message only for the first error encountered. The frequency of each type of error is accumulated, and execution is continued. MONERR may be invoked to alter these conventions, or to interrogate the error status, by the FORTRAN statement

$$JK = \text{MONERR}(KK,3),$$

where JK and KK are integers, related according to the following table. EC-N denotes the error count for error number N in the above table, and -- denotes no change from the previous setting.

<i>KK</i>	<i>JK</i>	<i>Error Counts</i>	<i>Subsequent Action Upon Finding Error</i>	
			<i>Print Msg.</i>	<i>Stop</i>
.LT.-1	0	--	--	--
-1	KK	--	Yes	Yes
0	KK	Reset to 0	1 Only	No
1	KK	--	Every Time	No
2	KK	--	No	No
3	*	--	--	--
4	EC-1	--	--	--
5	EC-2	--	--	--
.GT.5	0	--	--	--

* The return for KK=3 is the error number N (from the previous table) for the latest error encountered. At the same time, the internal error indicator is reset to 0. A return of 0 indicates no error has occurred.

COMMON BLOCKS

None

I/O

See "Special Conditions"

PRECISION

Single

REQUIRED ULIB
ROUTINES

MONERR

SPECIALIST

Margaret Drake, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

This is a routine in the Special Function Subroutine Package (FUNPACK) provided by the Applied Mathematics Division of Argonne National Laboratory.

ALGORITHM

The main computation involves evaluation of near minimax approximations published in Math. Comp. 19, 105-112 (1965) by W. J. Cody.

SPACE REQUIRED

 $1053_8 = 555_{10}$ locations

ACCURACY

This packet is designed to give near optimal accuracy in single precision arithmetic on a CDC 6000 series computer. The following table gives the results of random argument accuracy tests for arguments drawn from the interval (0, 1).

<i>Entry Point</i>	<i>Frequency of Bit Errors</i>		<i>MAX</i>	<i>RMS</i>
	<i>No. of Bits in Error</i>		<i>REL.</i>	<i>REL.</i>
	<i>0</i>	<i>1</i>	<i>ERROR</i>	<i>ERROR</i>
DELIPE	1787	213	6.760E-15	1.687E-15
DELIEL	1730	270	6.641E-15	1.799E-15
DELIEM	1803	197	6.949E-15	1.615E-15

TIMING

About 370 microseconds per evaluation

PORTABILITY

This version of FUNPACK has been specially tailored for the CDC 6000 series computer. If one desires to use these routines on a different computer, versions tailored for other machines are available from Argonne National Laboratory.

**REQUIRED RESIDENT
ROUTINES**

ABS, ALOG

LATEST REVISION September 1972

PURPOSE This packet computes single precision values of the complete elliptic integral of the first kind:

$$K(k^2) = \int_0^{\frac{1}{2}\pi} [1 - k^2 \sin^2 x]^{-\frac{1}{2}} dx \quad \text{for } 0 \leq k^2 \leq 1$$

ACCESS CARDS *FORTRAN,S=ULIB,N=ELIPK
 *COSY
 *FORTRAN,S=ULIB,N=MONERR
 *COSY

USAGE For maximum accuracy K may be computed as K(CAYSQ),
K(CAY**2), or K(1-ETA) by invoking the FUNCTION subprograms
 Y = DELIPK(CAYSQ) when $0 \leq \text{CAYSQ} < 1$,
 Y = DELIKL(CAY) when $-1 < \text{CAY} < 1$,
 Y = DELIKM(ETA) when $0 < \text{ETA} \leq 1$,
 respectively.

ARGUMENTS Only the self-explanatory ones noted above.

ENTRY POINTS

NATSEK, DELIPK, DELIK1, DELIKM, MONERR, FCNMON

SPECIAL CONDITIONS

The following table indicates the types of error that may be encountered in this routine and the function value supplied in each case (XIND is -1. -- an impossible function value).

<i>Error No.</i>	<i>Error</i>	<i>Function Values For</i>	
		<i>DELIPK or</i>	<i>DELIK1</i>
1	K**2 .LT. 0, or ETA .GT. 1	XIND	--
2	K**2 .EG. 1, or ETA .EQ. 0	XIND	XIND
3	K**2 .GT. 1, or ETA .LT. 0	XIND	XIND

All error processing is handled through FCNMON. The default error procedure is to print an error message only for the first error encountered. The frequency of each type of error is accumulated, and execution is continued. MONERR may be invoked to alter these conventions, or to interrogate the error status by the FORTRAN statement

$$JK = \text{MONERR}(KK, 2),$$

where JK and KK are integers, related according to the following table. EC-N denotes the error count for error number N in the above table, and -- denotes no change from the previous setting.

*Subsequent Action Upon
Finding Error*

<i>KK</i>	<i>JK</i>	<i>Error Counts</i>	<i>Print Mssg.</i>	<i>Traceback and Stop</i>
.LT.-1	0	--	--	--
-1	KK	--	Yes	Yes
0	KK	Reset to 0	1 Only	No
1	KK	--	Every Time	No
2	KK	--	No	No
3	*	--	--	--
4	EC-1	--	--	--
5	EC-2	--	--	--
6	EC-3	--	--	--
.GT.6	0	--	--	--

* The return for KK=3 is the error number N (from the previous table) for the latest error encountered. At the same time, the internal error indicator is reset to 0. A return of 0 indicates no error has occurred.

COMMON BLOCKS

None

I/O

See "Special Conditions"

PRECISION

Single

REQUIRED ULIB
ROUTINES

MONERR

SPECIALIST

Margaret Drake, NCAR, Boulder, Colorado 80303

7. ELIPK.4

LANGUAGE

FORTRAN

HISTORY

This is a routine in the Special Function Subroutine Package (FUNPACK) provided by the Applied Mathematics Division of Argonne National Laboratory.

ALGORITHM

The main computation involves evaluation of near minimax approximations published in Math. Comp. 19, 105-112 (1965) by W. J. Cody.

SPACE REQUIRED

$1066_8 = 566_{10}$ locations

ACCURACY

This subroutine is designed to give near optimal accuracy in single precision arithmetic on a CDC 6000 series computer. The following table gives the results of random argument accuracy tests for arguments drawn from the interval (0, 1).

<i>Entry Point</i>	<i>Frequency of Bit Errors</i>			<i>MAX</i>	<i>RMS</i>
	<i>No. of Bits in Error</i>			<i>REL.</i>	<i>REL.</i>
	<i>0</i>	<i>1</i>	<i>2</i>	<i>ERROR</i>	<i>ERROR</i>
DELIPK	1247	751	2	8.906E-15	2.956E-15
DELIK1	1243	757	0	7.104E-15	2.842E-15
DELIKM	1238	762	0	7.104E-15	2.954E-15

TIMING

About 360 microseconds per evaluation

PORTABILITY

This version of FUNPACK has been specially tailored for the CDC 6000 series computer. If one desires to use these routines on a different computer, versions tailored for other machines are available from Argonne National Laboratory.

**REQUIRED RESIDENT
ROUTINES**

ALOG

ARGUMENTS

Only the self-explanatory ones noted above.

ENTRY POINTS

NATSEI, DEI, DPEONE, DEXPEI, MONERR, FCNMON

SPECIAL CONDITIONS

The following table indicates the types of error that may be encountered in this routine and the function value supplied in each case (XINF is the largest positive machine number).

<i>Error No.</i>	<i>Error</i>	<i>Argument Range</i>	<i>Function Values For</i>		
			<i>DEI</i>	<i>DEXPEI</i>	<i>DPEONE</i>
1	UNDERFLOW	X .LT. XMIN	0	-	0
2	OVERFLOW	X .GE. XMAX	XINF	-	XINF
3	ILLEGAL X	X = 0	-XINF	-XINF	XINF
4	ILLEGAL X	X .LT. 0	-	-	USE ABS(X)

All error processing is handled through FCNMON. The default error procedure is to print an error message only for the first error encountered. The frequency of each type of error is accumulated, and execution is continued. MONERR may be invoked to alter these conventions, or to interrogate the error status by the FORTRAN statement

$$JK = \text{MONERR}(KK,1),$$

where JK and KK are integers, related according to the following table. EC-N denotes the error count for error number N in the above table, and -- denotes no change from the previous setting.

<i>KK</i>	<i>JK</i>	<i>Error Counts</i>	<i>Subsequent Action Upon Finding Error</i>	
			<i>Print Mssg.</i>	<i>Stop</i>
.LT.-1	0	--	--	--
-1	KK	--	Yes	Yes
0	KK	Reset to 0	1 Only	No
1	KK	--	Every Time	No
2	KK	--	No	No
3	*	---	---	---
4	EC-1	---	---	---
5	EC-2	---	---	---
6	EC-3	---	---	---
7	EC-4	---	---	---
.GT. 7	0	---	---	---

* The return for KK=3 is the error number N (from the previous table) for the latest error encountered. At the same time, the internal error indicator is reset to 0. A return of 0 indicates no error has occurred.

COMMON BLOCKS

None

I/O

See "Special Conditions"

PRECISION

Single

REQUIRED ULIB
ROUTINES

MONERR

SPECIALIST

Margaret Drake, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY This is a routine in the Special Function Subroutine Package (FUNPACK) provided by the Applied Mathematics Division of Argonne National Laboratory.

ALGORITHM The main computation involves evaluation of rational Chebyshev approximations published in Math. Comp. 22, 641-649 (1968), and Math. Comp. 289-303 (1969) by Cody and Thacher.

SPACE REQUIRED 1637₈ = 927₁₀ locations

ACCURACY This subroutine is designed to give near optimal accuracy in single precision arithmetic on a CDC 6000 series computer. The following table gives the results of random argument accuracy tests for the DEI entry.

<i>Argument Range</i>	<i>Frequency of Bit Errors No. of Bits in Error</i>					<i>MAX REL. ERROR</i>	<i>RMS REL. ERROR</i>
	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>		
(-50.0, -4.0)	845	989	166	0	0	1.430E-14	4.612E-15
(-4.0, -1.0)	734	943	316	7	0	2.032E-14	5.612E-15
(-1.0, 0.0)	957	888	155	0	0	1.414E-14	4.349E-15
(0.0, 5.0)	804	840	299	55	2	5.432E-14	7.207E-15
(6.0, 12.0)	626	781	427	154	12	4.733E-14	9.359E-15
(12.0, 24.0)	662	1060	278	0	0	1.575E-14	5.269E-15
(24.0, 100.)	1143	818	39	0	0	1.373E-14	3.541E-15

TIMING

About 440 microseconds for each evaluation

PORTABILITY

This version of FUNPACK has been specifically tailored for the CDC 6000 series computer. If one desires to use these routines on a different computer, versions tailored for other machines are available from Argonne National Laboratory.

**REQUIRED RESIDENT
ROUTINES**

ALOG, EXP

HYPER**PACKAGE HYPER****LATEST REVISION** April 1974**PURPOSE** HYPER is a package of the 3 hyperbolic functions: COSH, SINH, and TANH.**NOTE** TANH is also the name of a resident routine.**ACCESS CARDS** *FORTRAN,S=ULIB,N=HYPER
*COSY**USAGE** Y = TANH(X)
Y = SINH(X)
Y = COSH(X)**ARGUMENTS****On Input** X

Each function requires a real-valued argument.

7.HYPER.2

On Output

X
Unchanged.

ENTRY POINTS

TANH, SINH, COSH

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

N. Read, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

ALGORITHM

COSH: COSH(X) is evaluated, as defined, in terms of EXP(X).

SINH: For ABS(X) ≤ 0.5 , a rational approximation is employed.
For ABS(X) > 0.5 , SINH(X) is evaluated, as defined,
in terms of EXP(X).

TANH: TANH(X) is computed as SINH(X)/COSH(X).

SPACE REQUIRED $23_8 + 56_8 + 70_8 = 149_8$ locations

REFERENCE Hart, 1968: *Computer Approximations*. John Wiley & Sons, Inc., 96-104.

ACCURACY COSH: A maximum relative error of $\sim 1.5E-14$.
 SINH: A maximum relative error of $\sim 1.5E-14$.
 TANH: A maximum relative error of $\sim 3.0E-14$.

TIMING COSH: ~ 12 milliseconds per entry on CDC 7600.
 SINH: ~ 11 milliseconds per entry on CDC 7600.
 TANH: ~ 15 milliseconds per entry on CDC 7600.

PORTABILITY Standard FORTRAN used.

**REQUIRED RESIDENT
ROUTINES** EXP

8

FAST FOURIER TRANSFORMS

FFT

FFTPW2



PACKAGE FFT**PURPOSE**

Fast Fourier transforms for data of arbitrary length. The file FFT contains three routines to handle various forms of input data as tabulated below:

<u>Routine Name</u>	<u>Input Form</u>	<u>Output Form</u>
FFTRC	Real	Half Complex
FFTCR	Half Complex	Real
FFTCC	Complex	Complex

Half complex refers here to the first $N/2+1$ complex values of a conjugate symmetric array of length N .

ACCESS CARDS

*FORTRAN,S=ULIB,N=FFT
*COSY

ENTRY POINTS

FFTRC, FFTCR, FFTCC, FOURT

SPECIAL CONDITIONS

The efficiency of these routines is greatly affected by the number of points to be transformed. If N is the length of the transform (NRLPTS in FFTRC or FFTCR; NCPTS in FFTCC) the execution time is roughly proportional to $N \cdot \text{SUMPF}$ where SUMPF is the sum of the prime factors of N . The following table illustrates the advantage of choosing N to contain only small primes. Error is measured by transforming random data and reconstructing it using the inverse transform.

SPECIAL CONDITIONS
(continued)

*FFT Performance
on the
CDC 7600*

<i>N</i>	<i>Prime Factorization</i>	<i>Timing (milliseconds)</i>	<i>Relative Error</i>
23	23	1.2	1.E-12
25	5 ²	1.0	1.E-13
48	2 ⁴ ·3	1.8	6.E-14
100	2 ² ·5 ²	4.6	1.E-13
101	101	17.0	3.E-10
243	3 ⁵	16.0	1.E-13
512	2 ⁹	13.0	3.E-13
1008	2 ⁴ ·3 ² ·7	60.0	3.E-13
1009	1009	1597.0	3.E-7
2187	3 ⁷	192.0	2.E-13
7000	2 ⁴ ·5 ⁴ ·7	712.0	3.E-13
8191	8191	105300.0	1.E-6
10000	2 ⁵ ·5 ⁵	781.0	3.E-13

If N is a power of two, the package FFTPOW2 provides even greater efficiency.

SPACE REQUIRED

2544₈

SUBROUTINE FFTRC (RLDAT,NRLPTS,SIGNEX,HCTRN,NHCPTS,WORK,LWRK,IERR)**DIMENSION OF ARGUMENTS**

REAL RLDAT (NRLPTS)
 COMPLEX HCTRN (NHCPTS) where NHCPTS = NRLPTS/2+1
 REAL WORK (LWRK)

LATEST REVISION

November 1973

PURPOSE

Fourier transform (real to half complex). (In case SIGNEX=+1., the computation performed by this routine is sometimes called a forward transform or Fourier analysis.) The discrete Fourier transform of an array RLDAT, containing NRLPTS real values, is a set CF of NRLPTS complex values satisfying the conjugate symmetry relation $CF(NRLPTS+2-K) = CONJG(CF(K))$ ($K=2,NRLPTS$). Due to this symmetry relation, it is only necessary to compute the first NHCPTS = NRLPTS/2+1 complex values, and these are returned in the complex array HCTRN. For more detail, see the description of the argument HCTRN and the "Interpretation" section.

USAGE

CALL FFTRC (RLDAT,NRLPTS,SIGNEX,HCTRN,NHCPTS,WORK,LWRK,IERR)

The original values of RLDAT may be regenerated from HCTRN by first dividing all values of HCTRN by NRLPTS and then calling FFTCR (HCTRN,NHCPTS,-SIGNEX,RLDAT,NRLPTS,WORK,LWRK,IERR).

NOTE

For these comments we assume $SIGNEX=+1.$ or $-1.$ and define the complex function

$$CEX(X) = \exp\left(\frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot X}{NRLPTS}\right) \quad (\text{for all real } X)$$

where $\pi=3.14\dots$ and $i=\sqrt{-1}$.

An overscore, $\overline{\quad}$, will denote complex conjugation.

ARGUMENTS

On Input

RLDAT

A real array containing the NRLPTS data values to be transformed. It may be equivalenced to HCTRN or WORK if desired. The dimension is assumed to be $RLDAT(NRLPTS)$.

NRLPTS

The number of real data values to be transformed. For the greatest efficiency, it should be a product of small primes.

SIGNEX

A variable whose sign determines the sign of the argument of the complex exponential used in the transform computations. For convenience, we assume in these comments that $SIGNEX$ is $+1.$ or $-1.$, but the routine in fact only uses the sign of its value.

NHCPTS

The number of complex values to be returned as the half complex transform result in array HCTRN. It must be $=NRLPTS/2+1$ or a fatal error is flagged.

Note: NHCPTS is not an output parameter. It must be set to $NRLPTS/2+1$ by the user, and NHCPTS complex locations must be provided for the output array HCTRN.

WORK

A workspace of length LWRK (.GE.4*NRLPTS) for use by the routine. Either or both of the arrays RLDAT and HCTRN may be equivalenced to WORK to reduce storage. If RLDAT is so equivalenced, it will be destroyed.

LWRK

The length of array WORK. It must be .GE.4*NRLPTS or a fatal error is flagged.

On Output**HCTRN**

The complex array containing essentially the first half of the conjugate symmetric transform result. HCTRN(K) (for K=1,NHCPTS) is the complex value defined by

$$\text{HCTRN}(K) = \sum_{J=1}^{\text{NRLPTS}} \text{RLDAT}(J) \cdot \text{CEX}((J-1) \cdot (K-1))$$

These values are also referred to as the Fourier coefficients--see "Interpretation" section. The dimension is assumed to be COMPLEX HCTRN (NHCPTS) which requires 2*NHCPTS core locations.

Note: HCTRN may be equivalenced to RLDAT if desired, but they are not identical in size. The number of core locations required for HCTRN is NRLPTS+1 for odd NRLPTS and NRLPTS+2 for even NRLPTS.

WORK

The workspace containing intermediate results.

IERR

An error flag with the following meanings

- = 0 No error.
- = 101 NRLPTS is less than 1.
- = 102 NHCPTS is not NRLPTS/2+1.
- = 103 Insufficient workspace has been provided; LWRK is less than 4*NRLPTS.

8.FFT.6

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

SPECIALIST

Dave Fulker, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized March 1974 by Dave Fulker at NCAR.

ALGORITHM

A mixed radix fast Fourier transform. See, for example, Gentleman, W. M. and G. Sande, *Fast Fourier Transforms-- For Fun and Profit*, AFIPS Conference Proceedings, Vol. 29, 1966.

ACCURACY

Upon transforming and reconstructing an 10000 point random array, 11 digits are preserved. The accuracy is less if NRLPTS has large prime factors.

TIMING

The execution time is roughly proportional to $NRLPTS \cdot SUMPF$ where $SUMPF$ is the sum of the prime factors of $NRLPTS$. The proportionality factor is about 5 microseconds (CDC 7600).

PORTABILITY

Portable.

REQUIRED RESIDENT ROUTINES

ULIBER

An error printing routine

SIN, COS

Interpretation

A useful (equivalent) definition of the result of this routine is as follows

$$\begin{aligned}
 HCTR(K) = & \sum_{J=1}^{NRLPTS} RLDAT(J) \cdot \cos(2 \cdot \pi \cdot (J-1) \\
 & \cdot (K-1)/NRLPTS)) + i \cdot \text{SIGNEX} \cdot \sum_{J=1}^{NRLPTS} RLDAT(J) \\
 & \cdot \sin(2 \cdot \pi \cdot (J-1) \cdot (K-1)/NRLPTS)
 \end{aligned}$$

Assume F is some periodic function with period P . If $RLDAT$ represents F in the following sense

$$RLDAT(J) = F(P \cdot (J-1)/NRLPTS)$$

then $HCTR(K)$ is a trapezoidal rule approximation to

$$\begin{aligned}
 & \frac{NRLPTS}{P} \int_0^P F(X) \cos \frac{2 \cdot \pi \cdot (K-1) \cdot X}{P} dx \\
 & + i \cdot \text{SIGNEX} \cdot \frac{NRLPTS}{P} \int_0^P F(X) \sin \frac{2 \cdot \pi \cdot (K-1) \cdot X}{P} dx
 \end{aligned}$$

Interpretation
(continued)

See Isaacson, Eugene and Herbert Bishop Keller, *Analysis of Numerical Methods*, pp. 340-341 for an error analysis of the trapezoidal rule for periodic functions.

It is useful to note that HCTRN(1) is always real, and HCTRN(NHCPTS) is real if NRLPTS is even.

The transform results (Fourier coefficients) from this routine represent the data in the following sense

$$RLDAT(J) = \sum_{K=1}^{NRLPTS} CF(K) \cdot CEX(-(J-1) \cdot (K-1))$$

(for $J=1, NRLPTS$) where

$$CF(K) = \begin{cases} \frac{HCTRN(K)}{NRLPTS} & \text{for } 1 \leq K \leq NHCPTS \\ \frac{HCTRN(NRLPTS+2-K)}{NRLPTS} & \text{for } NHCPTS < K < NRLPTS \end{cases}$$

This computation may be performed by calling SUBROUTINE FFTCR (CF, NHCPTS, -SIGNEX, RLDAT, NRLPTS, WORK, LWRK, IERR) though only the first NHCPTS values of CF are used.

To obtain real interpolation, differentiation, or spectra, it is necessary to use the following representation (which is equivalent to the above for integer J)

$$RLDAT(J) = \sum_{K=-NILQ}^{NILQ} CF(K) \cdot CEX(-(J-1) \cdot K)$$

where

$$NILQ = \begin{cases} \frac{NRLPTS-1}{2} & \text{for odd NRLPTS} \\ \frac{NRLPTS}{2} & \text{for even NRLPTS} \end{cases}$$

and

$$CF(K) = \begin{cases} \frac{HCTRN(K+1)}{NRLPTS} & \text{for } 0 < K < \frac{NRLPTS}{2} \\ \frac{HCTRN(|K|+1)}{2 \cdot NRLPTS} & \text{for } K = \pm \frac{NRLPTS}{2} \quad (\text{used only if NRLPTS is even}) \\ \frac{HCTRN(-(K+1))}{NRLPTS} & \text{for } -\frac{NRLPTS}{2} < K < 0 \end{cases}$$

This representation can be used to derive the following sine and cosine representation

$$RLDAT(J) = A(0) + \sum_{J=1}^{NILQ} A(K) \cdot \cos \frac{2 \cdot \pi \cdot (J-1) \cdot K}{NRLPTS} \\ + B(K) \cdot \sin \frac{2 \cdot \pi \cdot (J-1) \cdot K}{NRLPTS}$$

where

$$A(K) = \begin{cases} \frac{\text{real}(HCTRN(1))}{NRLPTS} & \text{for } K=0 \\ \frac{2 \cdot \text{real}(HCTRN(K+1))}{NRLPTS} & \text{for } 0 < K < \frac{NRLPTS}{2} \\ \frac{\text{real}(HCTRN(K+1))}{NRLPTS} & \text{for } K = \frac{NRLPTS}{2} \quad (\text{used only if NRLPTS is even}) \end{cases}$$

and

$$B(K) = \begin{cases} \frac{\text{SIGNEX} \cdot 2 \cdot \text{imag}(HCTRN(K+1))}{NRLPTS} & \text{for } 0 < K < \frac{NRLPTS}{2} \\ 0 & \text{for } K = \frac{NRLPTS}{2} \quad (\text{used only if NRLPTS is even}) \end{cases}$$

SUBROUTINE FFTCR (HCDAT,NHCPTS,SIGNEX,RLTRN,NRLPTS,WORK,LWRK,IERR)**DIMENSION OF ARGUMENTS**

COMPLEX HCDAT (NHCPTS)
 REAL RLTRN (NRLPTS) where $NRLPTS/2+1=NHCPTS$
 REAL WORK (LWRK)

LATEST REVISION

November 1973

PURPOSE

Fourier transform (half complex to real). (In case $SIGNEX=-1.$, the computation performed by this routine is sometimes called a backward transform or Fourier synthesis.) HCDAT is assumed to be the first $NHCPTS=NRLPTS/2+1$ complex values of a set CF containing NRLPTS complex values which satisfy the conjugate symmetry relation $CF(NRLPTS+2-K)=CONJG(CF(K))$ ($K=2,NRLPTS$). In addition, HCTR(1) is assumed to be real. The discrete Fourier transform of such a conjugate symmetric array is a set of NRLPTS real values which are returned in the real array RLTRN. For more detail, see the description of the argument RLTRN and the "Interpretation" section of routine FFTRC.

USAGE

CALL FFTCR (HCDAT,NHCPTS,SIGNEX,RLTRN,NRLPTS,WORK,LWRK,IERR)

The original values of HCDAT may be regenerated from RLTRN by first dividing all values of RLTRN by NRLPTS and then calling FFTRC (RLTRN,NRLPTS,-SIGNEX,HCDAT,NHCPTS,WORK,LWRK,IERR).

NOTE

For these comments we assume $\text{SIGNEX}=+1.$ or $-1.$ and define the complex function

$$\text{CEX}(X) = \exp \frac{\text{SIGNEX} \cdot 2 \cdot \pi \cdot i \cdot X}{\text{NRLPTS}} \quad (\text{for all real } X)$$

where $\pi=3.14\dots$ and $i=\sqrt{-1}$.

An overscore $\overline{\quad}$ will denote a complex conjugation.

ARGUMENTS

On Input

HCDAT

A complex array containing NHCPTS complex values which comprise essentially the first half of the conjugate symmetric data to be transformed. It may be equivalenced to RLTRN or WORK if desired, but it is not of identical size.

The dimension is assumed to be $\text{COMPLEX HCDAT}(\text{NHCPTS})$ which requires $2 \cdot \text{NHCPTS}$ core locations.

NHCPTS

The number of complex values entered in HCDAT.

SIGNEX

A variable whose sign determines the sign of the argument of the complex exponential used in the transform computations. For convenience we assume in these comments that SIGNEX is $+1.$ or $-1.$, but the routine in fact only uses the sign of its value.

NRLPTS

The number of real transform values to be returned. NHCPTS must be $\text{NRLPTS}/2+1$ or a fatal error is flagged.

Note: NRLPTS is not an output parameter. It must satisfy $\text{NRLPTS}/2+1=\text{NHCPTS}$, and NRLPTS real locations must be provided for the output array RLTRN.

On Input
(continued)

WORK

A workspace of length LWRK (.GE.4*NRLPTS) for use by the routine. Either or both of the arrays HCDAT or RLTRN may be equivalenced to WORK to reduce storage. If HCDAT is so equivalenced, it will be destroyed.

LWRK

The length of array WORK. It must be .GE.4*NRLPTS or a fatal error is flagged.

On Output**RLTRN**

The real array containing the transform result. RLTRN(J) (for J=1,NRLPTS) is the real value defined by

$$RLTRN(J) = \sum_{K=1}^{NRLPTS} CF(K) \cdot CEX(J-1) \cdot (K-1)$$

where

$$CF(K) = \begin{cases} HCDAT(K) & \text{for } 1 \leq K \leq NHCPTS \\ \overline{HCDAT(NRLPTS+2-K)} & \text{for } NHCPTS < K < NRLPTS \end{cases}$$

The dimension is assumed to be RLTRN(NRLPTS).

Note: RLTRN may be equivalenced to HCDAT if desired.

WORK

The workspace containing intermediate results.

IERROR

An error flag with the following meanings:

- 0 No error.
- 101 NRLPTS is less than 1.
- 102 NHCPTS is not NRLPTS/2+1.
- 103 Insufficient workspace has been provided:
LWRK is less than 4*NRLPTS.

COMMON BLOCKS	None
I/O	None
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Dave Fulker, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Standardized March 1974 by Dave Fulker at NCAR.
ACCURACY	Upon transforming and reconstructing an 10000 point random array, 11 digits are preserved. The accuracy is less if NRLPTS has large prime factors.
TIMING	The execution time is roughly proportional to $NRLPTS * SUMPF$ where SUMPF is the sum of the prime factors of NRLPTS. The proportionality factor is about 5 microseconds on the CDC 7600.
PORTABILITY	Portable

REQUIRED RESIDENT
ROUTINES

ULIBER

An error printing routine.

SIN, COS

Interpretation

A useful (equivalent) definition of the result of this routine is as follows. Define

$$A(K) = \begin{cases} \frac{\text{real}(\text{HCTRN}(1))}{\text{NRLPTS}} & \text{for } K=0 \\ \frac{2 \cdot \text{real}(\text{HCTRN}(K+1))}{\text{NRLPTS}} & \text{for } 0 < K < \frac{\text{NRLPTS}}{2} \\ \frac{\text{real}(\text{HCTRN}(K+1))}{\text{NRLPTS}} & \text{for } K = \frac{\text{NRLPTS}}{2} \quad (\text{used only if } \text{NRLPTS} \text{ is even}) \end{cases}$$

$$B(K) = \begin{cases} \frac{2 \cdot \text{imag}(\text{HCTRN}(K+1))}{\text{NRLPTS}} & \text{for } 0 < K < \frac{\text{NRLPTS}}{2} \\ 0 & \text{for } K = \frac{\text{NRLPTS}}{2} \end{cases}$$

and

$$\text{NILQ} = \begin{cases} \frac{\text{NRLPTS}-1}{2} & \text{for odd NRLPTS} \\ \frac{\text{NRLPTS}}{2} & \text{for even NRLPTS} \end{cases}$$

Then

$$\begin{aligned} \text{RLDAT}(J) = & A(0) + \sum_{K=1}^{\text{NILQ}} A(K) \cos \frac{2 \cdot \pi \cdot (J-1) \cdot K}{\text{NRLPTS}} \\ & + \text{SIGNEX} \cdot \sum_{K=1}^{\text{NILQ}} B(K) \sin \frac{2 \cdot \pi \cdot (J-1) \cdot K}{\text{NRLPTS}} \end{aligned}$$

See the "Interpretation" section of the FFTRC write-up for further discussion.

SUBROUTINE FFTCC (CDATA,NCPTS,SIGNEX,CTRAN,WORK,IERR)**DIMENSION OF
ARGUMENTS**

COMPLEX CDATA (NCPTS),CTRAN (NCPTS)
REAL WORK(2*NCPTS)

LATEST REVISION

October 1973

PURPOSE

Fourier transform (complex to complex). (In case SIGNEX=+1., the computation performed by this routine is sometimes called a forward transform or Fourier analysis. For SIGNEX=-1., it is called a backward transform or Fourier synthesis.) The discrete Fourier transform of an array CDATA, containing NCPTS complex values, is a set of NCPTS complex values which are returned in the complex array CTRAN. For more detail, see the description of the argument CTRAN and the "Interpretation" section.

USAGE

CALL FFTCC (CDATA,NCPTS,SIGNEX,CTRAN,WORK,IERR)

The original values of CDATA may be regenerated from CTRAN by first dividing all values of CTRAN by NCPTS and then calling FFTCC (CTRAN,NCPTS,-SIGNEX,CDATA,WORK,IERR)

NOTE

For these comments we assume SIGNEX=+1. or -1. and define the complex function

$$CEX(X) = \exp \frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot X}{NCPTS} \text{ (for all real } X)$$

where $\pi=3.14\dots$ and $i=\sqrt{-1}$.

ARGUMENTS

On Input

CDATA

A complex array containing the NCPTS complex data values to be transformed. It may be equivalenced to WORK if desired. The dimension is assumed to be COMPLEX CDATA(NCPTS).

NCPTS

The number of complex data values to be transformed. It must be a positive power of 2 or a fatal error is flagged.

SIGNEX

A variable whose sign determines the sign of the argument of the complex exponential used in the transform computations. For convenience, we assume in these comments that SIGNEX is +1. or -1., but the routine in fact only uses the sign of its value.

WORK

A workspace of length 2*NCPTS for use by the routine. It may be equivalenced to CDATA in which case CDATA will be destroyed.

On Output

CTRAN

The complex array in which the NCPTS complex transform results are returned. CTRAN(K) (for K=1,NCPTS) is the complex value defined by

$$\text{CTRAN}(K) = \sum_{J=1}^{\text{NCPTS}} \text{CDATA}(J) \cdot \text{CEX}(J-1) \cdot (K-1)$$

These values are also referred to as the Fourier coefficients -- see the "Interpretation" section.

The dimension is assumed to be COMPLEX CTRAN(NCPTS).

WORK

The workspace containing intermediate results.

IERR

An error flag with the following meanings.

0 No error.

101 NCPTS is less than 1.

COMMON BLOCKS

None

I/O

None

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

SPECIALIST

Dave Fulker, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized March 1974 by Dave Fulker at NCAR.

ALGORITHM

The usual fast Fourier transform algorithm. See, for example, Gentleman, W. M. and G. Sande, *Fast Fourier Transforms -- For Fun and Profit*, AFIPS Conference Proceedings, Vol. 29, 1966.

ACCURACY

Upon transforming and reconstructing an 10000 point random array, 11 digits are preserved. The accuracy is less if NCPTS has large prime factors.

TIMING

The execution time is roughly proportional to NCPTS*SUMPF where SUMPF is the sum of the prime factors of NCPTS. The proportionality factor is about 5 microseconds on the CDC 7600.

PORTABILITY

Portable

REQUIRED RESIDENT ROUTINES

ULIBER

An error printing routine.

SIN, COS

Interpretation

Assume F is some periodic complex function with period P. If CDATA represents F in the following sense:

$$CDATA(J) = F(P \cdot (J-1)/NCPTS)$$

then CTRAN(K) is a trapezoidal rule approximation to

$$\frac{NCPTS}{P} \int_0^P F(X) \cdot \exp \frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot (K-1) \cdot X}{P} dx$$

See Isaacson, Eugene and Herbert Bishop Keller, *Analysis of Numerical Methods*, pp. 340-341 for an error analysis of the trapezoidal rule for periodic functions.

The transform results (Fourier coefficients) from this routine represent the data in the following sense:

$$CDATA(J) = \sum_{K=1}^{NCPTS} CF(K) \cdot CEX(-(J-1) \cdot (K-1)) \quad (\text{for } J=1, NCPTS)$$

where $CF(K) = CTRAN(K) / NCPTS$.

This computation may be performed by calling subroutine FFTCC (CF, NCPTS, -SIGNEX, CDATA, IERR). To obtain interpolation, differentiation, or spectra, it is necessary to use the following representation (which is equivalent to the above for integer J):

$$CDATA(J) = \sum_{K=-NILQ}^{NILQ} CF(K) \cdot CEX(-(J-1) \cdot K)$$

where

$$NILQ = \begin{cases} \frac{NCPTS-1}{2} & \text{for odd } NCPTS \\ \frac{NCPTS}{2} & \text{for even } NCPTS \end{cases}$$

$$CF(K) = \begin{cases} \frac{CTRAN(K+1)}{NCPTS} & \text{for } 0 < K < \frac{NCPTS}{2} \\ \frac{CTRAN(|K|+1)}{2 \cdot NCPTS} & \text{for } K = \pm \frac{NCPTS}{2} \quad (\text{used only if } NCPTS \text{ is even}) \\ \frac{CTRAN(NCPTS+K+1)}{NCPTS} & \text{for } -\frac{NCPTS}{2} < K < 0 \end{cases}$$

FFTPOW2

PURPOSE

Fast Fourier transforms for data whose length is a power of two. The file FFTPOW2 contains three routines to handle various forms of input data as tabulated below:

<u>Routine Name</u>	<u>Input Form</u>	<u>Output Form</u>
FFT2RC	Real	Half Complex
FFT2CR	Half Complex	Real
FFT2CC	Complex	Complex

Half complex refers here to the first $N/2+1$ complex values of a conjugate symmetric array of length N .

ACCESS CARDS

*FORTRAN,S=ULIB,N=FFTPOW2
 *COSY
 *ASCENT,S=ULIB,N=AFFT2
 *COSY

ENTRY POINTS

FFT2RC, FFT2CR, FFT2CC, FFT2

SPACE REQUIRED

1025₈ (including both files)

SUBROUTINE FFT2RC (RLDAT,NRLPTS,SIGNEX,HCTR,NHCPTS,IERROR)**DIMENSION OF ARGUMENTS**

REAL RLDAT (NRLPTS)
 COMPLEX HCTR (NHCPTS) where NHCPTS=NRLPTS/2+1

LATEST REVISION

November 1973

PURPOSE

Fourier transform for powers of 2 (real to half complex). (In case SIGNEX=+1., the computation performed by this routine is sometimes called a forward transform or Fourier analysis.) The discrete Fourier transform of an array RLDAT, containing NRLPTS (a power of 2) real values, is a set CF of NRLPTS complex values satisfying the conjugate symmetry relation $CF(NRLPTS+2-K) = CONJG(CF(K))$ ($K=2,NRLPTS$). Due to this symmetry relation it is only necessary to compute the first NHCPTS = NRLPTS/2+1 complex values, and these are returned in the complex array HCTR. For more detail, see the description of the argument HCTR and the "Interpretation" section.

USAGE

CALL FFT2RC (RLDAT,NRLPTS,SIGNEX,HCTR,NHCPTS,IERROR).

The original values of RLDAT may be regenerated from HCTR by first dividing all values of HCTR by NRLPTS and then calling FFT2CR (HCTR,NHCPTS,-SIGNEX,RLDAT,NRLPTS,IERROR).

NOTE

For these comments we assume $SIGNEX = +1.$ or $-1.$ and define the complex function

$$CEX(X) = \exp\left(\frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot X}{NRLPTS}\right) \quad (\text{for all real } X)$$

where $\pi = 3.14\dots$ and $i = \sqrt{-1}$.

An overscore $\overline{\quad}$ will denote complex conjugation.

ARGUMENTS**On Input****RLDAT**

A real array containing the NRLPTS data values to be transformed. It may be equivalenced to HCTRN if desired. The dimension is assumed to be RLDAT(NRLPTS).

NRLPTS

The number of real data values to be transformed. It must be a positive power of 2 or a fatal error is flagged.

SIGNEX

A variable whose sign determines the sign of the argument of the complex exponential used in the transform computations. For convenience we assume in these comments that SIGNEX is $+1.$ or $-1.$, but the routine in fact only uses the sign of its value.

NHCPTS

The number of complex values to be returned as the half complex transform result in array HCTRN. It must be $= NRLPTS/2+1$ or a fatal error is flagged.

Note: NHCPTS is not an output parameter. It must be set to $NRLPTS/2+1$ by the user, and NHCPTS complex locations must be provided for the output array HCTRN.

On Output

HCTRN

The complex array containing essentially the first half of the conjugate symmetric transform result. HCTRN(K) (for $K = 1, \text{NHCPTS}$) is the complex value defined by

$$\text{HCTRN}(K) = \sum_{J=1}^{\text{NRLPTS}} \text{RLDAT}(J) \cdot \text{CEX}((J-1) \cdot (K-1))$$

These values are also referred to as the Fourier coefficients--see "Interpretation" section. The dimension is assumed to be COMPLEX HCTRN (NHCPTS) which requires $2 \cdot \text{NHCPTS} = \text{NRLPTS} + 2$ core locations.

Note: HCTRN may be equivalenced to RLDAT if desired, but they are not identical in size.

IERROR

An error flag with the following meanings

0 No error.

101 NRLPTS is not a positive power of 2.

102 NHCPTS is not $\text{NRLPTS}/2 + 1$.

COMMON BLOCKS

FFT2CM

I/O

None

PRECISION

Single

REQUIRED ULIB
ROUTINES

FFT2

An assembly language implementation of the fast Fourier transform algorithm. The access cards are:

*ASCENT,S=ULIB,N=AFFT2

*COSY

A slower FORTRAN equivalent is available with access cards

*FORTRAN,S=ULIB,N=FFFT2

*COSY

SPECIALIST

Dave Fulker, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Developed 1971-1973 by Dave Fulker at NCAR; standardized November 9, 1973

ALGORITHM

A transform of NRLPTS real values may be computed by performing a single fully complex transform of length $\text{NRLPTS}/2$. See for example the appendices to NCAR manuscript 71-10, Orszag, Steven A., *Numerical Simulation of Incompressible Flows within Simple Boundaries*. I. Galerkin (Spectral) Representation, January 1971.

The computation of the complex transform is an Ascent (Assembly Language) version of the usual fast Fourier transform algorithm (radix 2). See for example Gentleman, W. M. and G. Sande, *Fast Fourier Transforms--For Fun and Profit*, AFIPS Conference Proceedings, Vol. 29, 1966.

ACCURACY

Upon transforming and reconstructing an 8192 point random array, 10 digits are preserved.

TIMING

The execution time is roughly proportional to $M \cdot \text{NRLPTS}$ where $2^{**}M = \text{NRLPTS}$. The proportionality factor is about 5 microseconds (CDC 6600) though it becomes smaller for large NRLPTS.

PORTABILITY

The FORTRAN version of FFT2 would ordinarily be required for use at other centers.

REQUIRED RESIDENT ROUTINES

ULIBER

An error printing routine

SIN, COS

INTERPRETATION

A useful (equivalent) definition of the result of this routine is as follows

$$\begin{aligned} \text{HCTR}(K) = & \sum_{J=1}^{\text{NRLPTS}} \text{RLDAT}(J) \cdot \cos(2 \cdot \pi \cdot (J-1) \cdot (K-1) / \text{NRLPTS}) \\ & + i \cdot \text{SIGNEX} \cdot \sum_{J=1}^{\text{NRLPTS}} \text{RLDAT}(J) \cdot \sin(2 \cdot \pi \cdot (J-1) \cdot (K-1) / \text{NRLPTS}) \end{aligned}$$

Assume F is some periodic function with period P . If RLDAT represents F in the following sense

$$\text{RLDAT}(J) = F(P \cdot (J-1) / \text{NRLPTS})$$

then $\text{HCTR}(K)$ is a trapezoidal rule approximation to

$$\frac{\text{NRLPTS}}{P} \int_0^P F(X) \cos\left(\frac{2 \cdot \pi \cdot (K-1) \cdot X}{P}\right) dx$$

$$+ i \cdot \text{SIGNEX} \cdot \frac{\text{NRLPTS}}{P} \int_0^P F(X) \sin\left(\frac{2 \cdot \pi \cdot (K-1) \cdot X}{P}\right) dx$$

See Isaacson, Eugene and Herbert Bishop Keller, *Analysis of Numerical Methods*, pp. 340-341 for an error analysis of the trapezoidal rule for periodic functions.

It is useful to note that HCTRN(1) and HCTRN(NHCPTS) are always real.

The transform results (Fourier coefficients) from this routine represent the data in the following sense

$$\text{RLDAT}(J) = \sum_{K=1}^{\text{NRLPTS}} \text{CF}(K) \cdot \text{CEX}(-(J-1) \cdot (K-1))$$

(For J=1, NRLPTS)

where

$$\text{CF}(K) = \begin{cases} \frac{\text{HCTRN}(K)}{\text{NRLPTS}} & \text{for } 1 \leq K \leq \text{NHCPTS} \\ \frac{\text{HCTRN}(\text{NRLPTS}+2-K)}{\text{NRLPTS}} & \text{for } \text{NHCPTS} < K < \text{NRLPTS} \end{cases}$$

This computation may be performed by calling SUBROUTINE FFT2CR (CF, NHCPTS, -SIGNEX, RL DAT, NRLPTS, IERROR), though only the first NHCPTS values of CF are used.

INTERPRETATION
(continued)

To obtain real interpolation, differentiation, or spectra, it is necessary to use the following representation (which is equivalent to the above for integer J)

$$RLDAT(J) = \sum_{K=-\frac{NRLPTS}{2}}^{\frac{NRLPTS}{2}} CF(K) \cdot CEX(-(J-1) \cdot K)$$

where

$$CF(K) = \begin{cases} \frac{HCTRN(K+1)}{NRLPTS} & \text{for } 0 \leq K < \frac{NRLPTS}{2} \\ \frac{HCTRN(|K|+1)}{2 \cdot NRLPTS} & \text{for } K = \pm \frac{NRLPTS}{2} \\ \frac{HCTRN(-K+1)}{NRLPTS} & \text{for } -\frac{NRLPTS}{2} < K < 0 \end{cases}$$

This representation can be used to derive the following sine and cosine representation

$$RLDAT(J) = A(0) + A\left(\frac{NRLPTS}{2}\right) \cdot \cos(\pi \cdot (J-1)) \\ + \sum_{K=1}^{\frac{NRLPTS}{2}-1} \left[A(K) \cdot \cos\left(\frac{2 \cdot \pi \cdot (J-1) \cdot K}{NRLPTS}\right) \right. \\ \left. + B(K) \cdot \sin\left(\frac{2 \cdot \pi \cdot (J-1) \cdot K}{NRLPTS}\right) \right]$$

where

$$A(K) = \begin{cases} \frac{\text{real}(HCTRN(K+1))}{NRLPTS} & \text{for } K=0 \text{ and} \\ & \text{for } K=\frac{NRLPTS}{2} \\ \frac{2 \cdot \text{real}(HCTRN(K+1))}{NRLPTS} & \text{for } 0 < K < \frac{NRLPTS}{2} \end{cases}$$

$$B(K) = \frac{\text{SIGNEX} \cdot 2 \cdot \text{imag}(\text{HCTRN}(K+1))}{\text{NRLPTS}} \text{ for } 0 < k < \frac{\text{NRLPTS}}{2}$$

SUBROUTINE FFT2CR (HCDAT, NHCPTS, SIGNEX, RLTRN, NRLPTS, IERROR)

DIMENSION OF ARGUMENTS COMPLEX HCDAT (NHCPTS)
REAL RLTRN (NRLPTS) where $NRLPTS/2+1 = NHCPTS$

LATEST REVISION November 1973

PURPOSE Fourier transform for powers of 2 (half complex to real).
(In case $SIGNEX=-1.$, the computation performed by this routine is sometimes called a backward transform or Fourier synthesis.) HCDAT is assumed to be the first $NHCPTS=NRLPTS/2+1$ complex values of a set CF containing NRLPTS (a power of 2) complex values which satisfy the conjugate symmetry relation $CF(NRLPTS+2-K)=CONJG(CF(K))$ ($K=2,NRLPTS$). In addition, HCTR(1) is assumed to be real. The discrete Fourier transform of such a conjugate symmetric array is a set of NRLPTS real values which are returned in the real array RLTRN. For more detail, see the description of the argument RLTRN and the "Interpretation" section of routine FFT2RC.

USAGE CALL FFT2CR (HCDAT, NHCPTS, SIGNEX, RLTRN, NRLPTS, IERROR)

The original values of HCDAT may be regenerated from RLTRN by first dividing all values of RLTRN by NRLPTS and then calling FFT2RC (RLTRN, NRLPTS, -SIGNEX, HCDAT, NHCPTS, IERROR).

NOTE

For these comments we assume $SIGNEX = +1.$ or $-1.$ and define the complex function

$$CEX(X) = \exp \frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot X}{NRLPTS} \quad (\text{for all real } X)$$

where $\pi = 3.14\dots$ and $i = \sqrt{-1}$.

An overscore $\overline{\quad}$ will denote complex conjugation.

ARGUMENTS**On Input****HCDAT**

A complex array containing $NHCPTS$ complex values which comprise essentially the first half of the conjugate symmetric data to be transformed. It may be equivalenced to $RLTRN$ if desired, but it is not of identical size.

The dimension is assumed to be $COMPLEX\ HCDAT(NHCPTS)$ which requires $2*NHCPTS=NRLPTS+2$ core locations.

NHCPTS

The number of complex values entered in $HCDAT$. $NRLPTS=(NHCPTS-1)*2$ must be a positive power of 2 or a fatal error is flagged.

SIGNEX

A variable whose sign determines the sign of the argument of the complex exponential used in the transform computations. For convenience we assume in these comments that $SIGNEX$ is $+1.$ or $-1.$, but the routine in fact only uses the sign of its value.

NRLPTS

The number of real transform values to be returned. It must be $(NHCPTS-1)*2$ or a fatal error is flagged.

Note: $NRLPTS$ is not an output parameter. It must be set to $(NHCPTS-1)*2$ by the user, and $NRLPTS$ real locations must be provided for the output array $RLTRN$.

On Output

RLTRN

The real array containing the transform result. RLTRN(J) (for J = 1, NRLPTS) is the real value defined by

$$RLTRAN(J) = \sum_{K=1}^{NRLPTS} CF(K) \cdot CEX((J-1) \cdot (K-1))$$

where

$$CF(K) = \begin{cases} HCDAT(K) & \text{for } 1 \leq K \leq NHCPTS \\ \overline{HCDAT(NRLPTS+2-K)} & \text{for } NHCPTS < K < NRLPTS \end{cases}$$

The dimension is assumed to be RLTRN(NRLPTS).

Note: RLTRN may be equivalenced to NCDAT if desired.

IERROR

An error flag with the following meanings:

- 0 No error.
- 101 NRLPTS is not a positive power of 2.
- 102 NHCPTS is not NRLPTS/2+1.

COMMON BLOCKS

FFT2CM

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

FFT2

An assembly language implementation of the fast Fourier transform algorithm. The access cards are:

*ASCENT,S=ULIB,N=AFFT2

*COSY

A slower FORTRAN equivalent is available with access cards:

*FORTRAN,S=ULIB,N=FFT2

*COSY

SPECIALIST

Dave Fulker, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Developed 1971 - 1973 by Dave Fulker at NCAR; standardized November, 1973.

ALGORITHM

A transform of NRLPTS conjugate symmetric values may be computed by performing a single fully complex transform of length NRLPTS/2. See, for example, the appendices to NCAR manuscript 71-10, Orszag, Steven A., *Numerical Simulation of Incompressible Flows within Simple Boundaries*, I. Galerkin (Spectral) Representations, January 1971.

The computation of the complex transform is an ASCENT (Assembly Language) version of the usual fast Fourier transform algorithm (radix 2). See, for example, Gentleman, W. M. and G. Sande, *Fast Fourier Transforms - For Fun and Profit*, AFIPS Conference Proceedings, Vol. 29, 1966.

ACCURACY

Upon transforming and reconstructing an 8192 point random array, 10 digits are preserved.

TIMING

The execution time is roughly proportional to $M \cdot \text{NRLPTS}$ where $2^{**}M = \text{NRLPTS}$. The proportionality factor is about 5 microseconds (CDC 6600) though it becomes smaller for large NRLPTS .

PORTABILITY

The FORTRAN version of FFFT2 would ordinarily be required for use at other centers.

REQUIRED RESIDENT ROUTINES

ULIBER

An error printing routine.

SIN,COS

INTERPRETATION

A useful (equivalent) definition of the result of this routine routine is as follows

$$\text{RLTRN}(J) = \text{real}(\text{HCTRN}(1)) + 2 \cdot \sum_{K=2}^{\text{NHCPTS}-1} \text{real}(\text{HCTRN}(K))$$

$$\cdot \text{COS}(2 \cdot \pi \cdot (K-1) \cdot (J-1) / \text{NRLPTS}) + \text{SIGNEX}$$

$$\cdot 2 \cdot \sum_{K=2}^{\text{NHCPTS}-1} \text{imag}(\text{HCTRN}(K)) \cdot \text{SIN}(2 \cdot \pi \cdot (K-1) \cdot (J-1) / \text{NRLPTS})$$

$$+ \text{real}(\text{HCTRN}(\text{NHCPTS})) \cdot \text{COS}(\pi \cdot (J-1))$$

See the comments in FFT2RC for further discussion.

SUBROUTINE FFT2CC (CDATA,NCPTS,SIGNEX,CTRAN,IERROR)

DIMENSION OF ARGUMENTS COMPLEX CDATA (NCPTS),CTRAN (NCPTS)

LATEST REVISION October 1973

PURPOSE Fourier transform for powers of 2 (complex to complex). (In case SIGNEX = +1., the computation performed by this routine is sometimes called a forward transform or Fourier analysis. For SIGNEX = -1., it is called a backward transform or Fourier synthesis.) The discrete Fourier transform of an array CDATA, containing NCPTS (a power of 2) complex values, is a set of NCPTS complex values which are returned in the complex array CTRAN. For more detail see the description of the argument CTRAN and the "Interpretation" section.

USAGE CALL FFT2CC (CDATA,NCPTS,SIGNEX,CTRAN,IERROR)

The original values of CDATA may be regenerated from CTRAN by first dividing all values of CTRAN by NCPTS and then calling FFT2CC (CTRAN,NCPTS,-SIGNEX,CDATA,IERROR).

NOTE For these comments we assume SIGNEX = +1. or -1. and define the complex function

$$CEX(X) = \exp\left(\frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot X}{NCPTS}\right) \text{ (for all real } X\text{)}$$

where $\pi = 3.14\dots$ and $i = \sqrt{-1}$

ARGUMENTS

On Input

CDATA

A complex array containing the NCPTS complex data values to be transformed. It may be equivalenced to CTRAN if desired.

The dimension is assumed to be COMPLEX CDATA(NCPTS).

NCPTS

The number of complex data values to be transformed. It must be a positive power of 2 or a fatal error is flagged.

SIGNEX

A variable whose sign determines the sign of the argument of the complex exponential used in the transform computations. For convenience we assume in these comments that SIGNEX is +1. or -1., but the routine in fact only uses the sign of its value.

On Output

CTRAN

The complex array in which the NCPTS complex transform results are returned.

$$\text{CTRAN}(K) = \sum_{J=1}^{\text{NCPTS}} \text{CDATA}(J) \cdot \text{CEX}((J-1) \cdot (K-1))$$

These values are also referred to as the Fourier coefficients -- see "Interpretation" section.

The dimension is assumed to be COMPLEX CTRAN(NCPTS).

Note: CTRAN may be equivalenced to CDATA if desired.

IERROR

An error flag with the following meanings:

0 No error.

101 NCPTS is not a positive power of 2

COMMON BLOCKS FFT2CM

I/O None

PRECISION Single

**REQUIRED ULIB
ROUTINES** FFFT2

 An assembly language implementation of the fast Fourier
 transform algorithm. The access cards are:

 *ASCENT,S=ULIB,N=AFFT2
 *COSY

 A slower FORTRAN equivalent is available with
 access cards

 *FORTRAN,S=ULIB,N=FFFT2
 *COSY

SPECIALIST Dave Fulker, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Developed 1971-1973 by Dave Fulker at NCAR, standardized
 October 16, 1973.

ALGORITHM This routine does little more than call FFFT2 which is an
 ASCENT (Assembly Language) version of the usual fast
 Fourier transform algorithm (radix 2). See, for example,
 Gentleman, W. M. and G. Sande, *Fast Fourier Transforms --
 For Fun and Profit*, AFIPS Conference Proceedings, Vol. 29, 1966.

ACCURACY

Upon transforming and reconstructing an 8192 point random array, 10 digits are preserved.

TIMING

The execution time is roughly proportional to $M \cdot \text{NCPTS}$ where $2 \cdot M = \text{NCPTS}$. The proportionality factor is about 7 microseconds (CDC 6600) though it becomes smaller for large NCPTS.

PORTABILITY

The FORTRAN version of FFFT2 and the elimination of LOCF calls would ordinarily be necessary for use at other centers.

REQUIRED RESIDENT ROUTINES

ULIBER

An error printing routine.

LOCF

A function whose value is the location in core of its argument. Its use may be avoided by removing the card (near statement with label 10) which reads:

```
IF (LOCF(CDATA).EQ.LOCF(CTRAN)) GO TO 20
```

Removal of this card will not affect the correctness of the routine, but will slow it slightly in case CDATA and CTRAN have been equivalenced.

SIN, COS

INTERPRETATION

Assume F is some periodic complex function with period P. If CDATA represents F in the following sense

$$\text{CDATA}(J) = F(P \cdot (J-1)/\text{NCPTS})$$

then CTRAN(K) is a trapezoidal rule approximation to

$$\frac{NCPTS}{P} \int_0^P F(X) \cdot \exp\left(\frac{SIGNEX \cdot 2 \cdot \pi \cdot i \cdot (K-1) \cdot X}{P}\right) dx$$

See Isaacson, Eugene and Herbert Bishop Keller, *Analysis of Numerical Methods*, pp. 340-341 for an error analysis of the trapezoidal rule for periodic functions.

The transform results (Fourier coefficients) from this routine represent the data in the following sense:

$$CDATA(J) = \sum_{K=1}^{NCPTS} CF(K) \cdot CEX(-(J-1) \cdot (K-1)) \quad (\text{for } J=1, NCPTS)$$

where $CF(K) = CTRAN(K)/NCPTS$.

This computation may be performed by calling subroutine FFT2CC (CF,NCPTS,-SIGNEX,CDATA,IERROR). To obtain interpolation, differentiation, or spectra it is necessary to use the following representation (which is equivalent to the above for integer J):

$$CDATA(J) = \sum_{K=-\frac{NCPTS}{2}}^{\frac{NCPTS}{2}} CF(K) \cdot CEX(-(J-1) \cdot K)$$

where

$$CF(K) = \begin{cases} \frac{CTRAN(K+1)}{NCPTS} & \text{for } 0 \leq K < \frac{NCPTS}{2} \\ \frac{CTRAN(|K|+1)}{2 \cdot NCPTS} & \text{for } K = \pm \frac{NCPTS}{2} \\ \frac{CTRAN(NCPTS+K+1)}{NCPTS} & \text{for } -\frac{NCPTS}{2} < K < 0 \end{cases}$$

STATISTICS/RANDOM NUMBERS

CNFPRB

RGRSN1

RGRSN2

RGRSN3

RGRSN4

RGRSN5

RGRSN6

RLHPTS

RNDEV

SPAL

SPECFT

FUNCTION CNFPRB (AWT, AH, GT, X, XHT, BWT, TEMP, MAWT, MAH, M, MGT, N, K, IP, ISW)

**DIMENSION OF
ARGUMENTS**

AWT(MAWT, N), AH(MAH, N), GT(MGT, K), X(N), XHT(N), BWT(M), IP(N),
TEMP(2*K+N)

LATEST REVISION

December 1973

PURPOSE

CNFPRB calculates the probability that a linear combination of an alternate solution vector, $G \cdot XHT$, is a credible replacement for the same linear combination of the calculated regression coefficients, $G \cdot X$.

ACCESS CARDS

*FORTRAN, S=ULIB, N=CNFPRB
*COSY

USAGE

PROB=CNFPRB (AWT, AH, GT, X, XHT, BWT, TEMP, MAWT, MAH, M, MGT, N, K, IP,
ISW)

ARGUMENTS

On Input

AWT

Contains the elements of the data matrix in the regression model.

AH

Contains the Householder decomposition of AWT as calculated in HDEC.

GT

Contains the transpose of the matrix of linear combinations.

X

Contains the calculated regression coefficients.

XHT

Contains the elements of the vector assumed to be another solution to the regression model.

BWT

Contains the observations of the random variables in the regression model.

TEMP

Is a vector used internally for working storage. It must have dimension at least $2*K+N$.

MAWT

Is an integer variable set equal to the row dimension of the matrix AWT as declared in the dimension statement of the calling program.

M

Is an integer variable set equal to the row dimension of the matrix AWT actually used in this function.

MGT

Is an integer variable set equal to the row dimension of the matrix GT as declared in the dimension statement of the calling program.

N

Is an integer variable set equal to the column dimension of AWT.

K

Is an integer set equal to the column dimension of the matrix GT.

IP

Is an integer input vector which contains information about the column permutations of the matrix AH as calculated in HDEC.

On Output

ISW

Is an integer error flag.
 = 1 if the matrix GT is singular.
 = 2 if $K \leq 0$ or $(M-N) \leq 0$.
 = 0 otherwise.

ENTRY POINTS

CNFPRB, SOLVLT, HDEC, FINIGL, GAUSS

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

If the matrix GT is singular, the function prints the message

G MATRIX ROW RANK DEFICIENT IN CNFPRB

If $K \leq 0$ or $(M-N) \leq 0$, the function prints the message

PARAMETERS OUT OF RANGE IN FINTGL

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized December 1973

ALGORITHM

The matrix AH contains the orthogonal decomposition of the matrix AWT such that

$$AWT = Q * \begin{bmatrix} I \\ \cdot \\ 0 \end{bmatrix} * R$$

where Q is $m \times m$ orthogonal, $\begin{bmatrix} I \\ \cdot \\ 0 \end{bmatrix} = \begin{bmatrix} 1. & 0 \\ 0 & \cdot \\ \cdot & 1 \\ 0 & \cdot \end{bmatrix}$, and

R is $n \times n$ upper triangular.

The matrix $(R^{-1})^T * GT$ is calculated and its transpose is stored in GT, destroying the original input matrix GT. This new matrix is decomposed such that

$$GT = S * \begin{bmatrix} I \\ \cdot \\ 0 \end{bmatrix} * Q$$

where S is $k \times k$ lower triangular, $\begin{bmatrix} I \\ \cdot \\ 0 \end{bmatrix} = \begin{bmatrix} 1. & 0 & \cdot \\ 0 & \cdot & 1 \\ \cdot & \cdot & \cdot \\ 0 & \cdot & 0 \end{bmatrix}$, and Q is $n \times n$ orthogonal.

The vector $Y = (G * X - G * XHT)$ is evaluated, and the lower triangular linear system, $S * Z = Y$, is solved for the vector Z.

The quantity

$$FSTAT = \frac{Z^T * Z}{\|AWT * X - BWT\|_2^2} * \frac{(M-N)}{K}$$

is the F-statistic for testing if $G * XHT$ is an acceptable replacement for $G * X$.

ALGORITHM
(continued)

The function value CNFPRB is

$$\text{CNFPRB} = 1. - \int_0^{\text{FSTAT}} h(f)df$$

where $h(f)$ is the F density function.

SPACE REQUIRED

$2021_8 = 1041_{10}$

ACCURACY

The assumption is made that the error vector for the regression model has a multivariate normal distribution with mean zero and covariance equal to $\sigma^2 I$. If this assumption is not satisfied, the F-statistic derived in CNFPRB will be inaccurate. For a discussion of the effects on this test, if the error does not behave as described above, see Plackett, *Regression Analysis*, (1960), p. 72.

TIMING

The timing is proportional to $KN^2 + MN$. For $M=10$, $N=6$, and $K=6$, CNFPRB takes 2 milliseconds on the CDC 7600.

PORTABILITY

CNFPRB is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, SQRTF, ATANF, RBAREX

SUBROUTINE RGRSN1 (A,MA,M,N,B,ITS,TEMP,X,IER)**DIMENSION OF ARGUMENTS**

A(MA,N),B(M),TEMP(N*(3+M)+M),X(N)

LATEST REVISION

December 1973

PURPOSE

RGRSN1 calculates a set of regression coefficients for the unweighted regression model $E(B) = A*x$, where E is the expectation operator.

ACCESS CARDS

*FORTRAN,S=ULIB,N=RGRSN1
*COSY

USAGE

CALL RGRSN1 (A,MA,M,N,B,ITS,TEMP,X,IER)

ARGUMENTS**On Input**

A

Is an input two dimensional variable with row dimension MA and column dimension N. On input, A contains the data matrix for regression analysis.

On Input
(continued)

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the actual number of rows of A used in the subroutine.

N

Is an integer input variable set equal to the column dimension of A.

B

Is a real input vector with dimension M. On input, B contains the elements of the right hand side.

ITS

Is an integer input variable set equal to the maximum number of iterations for iterative improvement.
(ITS.GE.0)

TEMP

Is a real variable used internally for working storage. It must have dimension, at least $N*(3+M)+M$.

On Output

X

Is a real output variable with dimension N. On output, X contains the calculated regression coefficients.

IER

Is an integer error flag.
= 1 if the matrix A is singular.
= 0 otherwise.

ENTRY POINTS

RGRSNI, HDEC, SOLVEH, HSITIM

COMMON BLOCKS None

I/O All messages are printed using ULIBER. If the matrix A is singular, the message

DATA MATRIX SINGULAR IN RGRSNI

is printed.

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Jo Walsh, NCAR, Boulder, Colorado 80303.

LANGUAGE FORTRAN

HISTORY Standardized December 1973

ALGORITHM The matrix A is orthogonally decomposed using subroutine HDEC such that

$A = Q * R$

where Q is $m \times m$ orthogonal and R is $m \times n$ upper triangular.

ALGORITHM
(continued)

Thus,

$$\begin{aligned} A*x &\approx B \\ (Q*R)*x &\approx B \\ R*x &\approx Q^T*B \end{aligned}$$

The first n rows of this overdetermined system form a square upper triangular linear system. This square system is solved for x using subroutine SOLVEH.

Iterative improvement is performed on the calculated regression coefficients using subroutine HSITIM.

SPACE REQUIRED1421₈ = 785₁₀**ACCURACY**

The accuracy depends on the conditioning of the matrix A and the error in the measurement of the vector B .

TIMING

The timing is proportional to the quantity $M*N^2$. For $M = 10$ and $N = 6$, RGRSN1 takes 2 milliseconds on the CDC 7600.

PORTABILITY

RGRSN1 is written in Standard FORTRAN. It has been run on the CDC 6600 and 7600 only.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, SQRTF

SUBROUTINE RGRSN2 (A,MA,M,N,B,ITS,TEMP,X,WT,IER)**DIMENSION OF
ARGUMENTS**

A(MA,N),B(M),X(N),WT(M),TEMP(2*N*(M+1)+M)

LATEST REVISION

December 1973

PURPOSE

RGRSN2

1. weights the regression model $E(B) = A*x$ with a diagonal matrix of weights whose diagonal elements are stored in WT and
2. calculates the regression coefficients of the weighted model.

ACCESS CARDS*FORTRAN,S=ULIB,N=RGRSN2
*COSY**USAGE**

CALL RGRSN2 (A,MA,M,N,B,ITS,TEMP,X,WT,IER)

ARGUMENTS

On Input

A

Is an input two dimensional variable with row dimension MA and column dimension N. On input, A contains the data matrix for regression analysis.

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the actual number of rows of A used in the subroutine.

N

Is an integer input variable set equal to the column dimension of A.

B

Is a real input vector with dimension M. On input, B contains the elements of the right hand side.

ITS

Is an integer input variable set equal to the maximum number of iterations for iterative improvement.
(ITS.GE.0)

TEMP

Is a real variable used internally for working storage. It must have dimension at least $2*N*(M+1)+M$.

WT

Is a real input variable with dimension M. On input, WT must contain the vector of weights.

On Output X
Contains the calculated regression coefficients of the weighted model.

IER
Is an integer error flag.
= 1 if the weighted data matrix is singular.
= 0 otherwise.

ENTRY POINTS RGRSN2, WIVEC, HDEC, SOLVEH, HSITIM

COMMON BLOCKS None

I/O All messages are printed using ULIBER.

If the weighted data matrix is singular, the message

WEIGHTED MATRIX SINGULAR IN RGRSN2

is printed.

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY

Standardized December 1973

ALGORITHM

The weighted regression model

$$AWT^*x = BWT$$

is calculated, where

$$AWT = D^*A$$

$$BWT = D^*B$$

and D is the diagonal matrix of weights whose diagonal elements are stored in the vector WT.

This weighted data matrix AWT is orthogonally decomposed using subroutine HDEC, such that

$$AWT = Q^*R$$

where Q is $m \times m$ orthogonal, and R is $n \times n$ upper triangular. Thus,

$$AWT^*x = B$$

$$(Q^*R)^*x = B$$

$$R^*x = Q^T B$$

The first n rows of this overdetermined system form an $n \times n$ upper triangular linear system which can be solved for x using subroutine SOLVEH.

Subroutine HSITIM performs iterative improvement on the calculated regression coefficients x.

SPACE REQUIRED

1716₈ = 974₁₀

ACCURACY

The accuracy depends on the conditioning of the weighted data matrix and the error in the measurement of the vector B.

TIMING

The timing is proportional to the quantity $MN^2 + MN$. For $M = 10$ and $N = 6$, RGRSN2 takes 2 milliseconds on the CDC 7600.

PORTABILITY

RGRSN2 is written in Standard FORTRAN. It has been run on the CDC 6600 and 7600 only.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, SQRTF

SUBROUTINE RGRSN3 (A,MA,M,N,B,ITS,TEMP,X,COV,MCOV,IER)

DIMENSION OF ARGUMENTS A(MA,N),B(M),TEMP(2*N*(M+1)+M*(M+5)),X(N),COV(MCOV,M)

LATEST REVISION December 1973

PURPOSE Given the regression model

$$A*X = b + \epsilon$$

where

- A is an $m \times n$ known data matrix,
- b is a set of observations of m random variables, and
- ϵ is an unknown vector of random errors with a known covariance matrix not equal to $\sigma^2 * I$,

then RGRSN3

1. weights the regression model using the known covariance matrix of ϵ , and
2. calculates the regression coefficients for the weighted model.

ACCESS CARDS *FORTRAN,S=ULIB,N=RGRSN3
 *COSY

USAGE

CALL RGRSN3 (A,MA,M,N,B,ITS,TEMP,X,COV,MCOV,IER)

ARGUMENTS

On Input

A

Is an input two dimensional variable with row dimension MA and column dimension N. On input, A contains the data matrix for regression analysis.

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the actual number of rows of A used in the subroutine.

N

Is an integer input variable set equal to the column dimension of A.

B

Is a real input vector with dimension M. On input, B contains the elements of the right hand side.

ITS

Is an integer input variable set equal to the maximum number of iterations for iterative improvement (ITS .GE. 0).

TEMP

Is a real variable used internally for working storage. It must have dimension at least $2*N*(M+1)+M*(M+5)$

COV

Is a real input two dimensional variable with row dimension MCOV and column dimension M. On input, COV must contain the covariance matrix of the right hand side.

MCOV

Is an integer input variable set equal to the row dimension of COV as declared in the dimension statement of the calling program.

On Output

X

Contains the calculated regression coefficients for the weighted regression model.

IER

Is an integer error flag.

= 1 if the matrix COV is singular

= 2 if the weighted data matrix is singular

= 0 otherwise.

ENTRY POINTS

RGRSN3, WIMTX, HDEC, SOLVEH, HSITIM, CHLSKY, CHSLV1, CHITIM

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

If the input covariance matrix COV is singular, the subroutine prints the message

COVARIANCE MATRIX SINGULAR IN RGRSN3.

If the weighted data matrix is singular, the subroutine prints the message

WEIGHTED MATRIX SINGULAR IN RGRSN3.

PRECISION Single

REQUIRED ULIB
ROUTINES None

SPECIALIST Jo Walsh, NCAR, Boulder, Colorado 80303.

LANGUAGE FORTRAN

HISTORY Standardized December 1973

ALGORITHM The covariance matrix COV is decomposed into the product of a lower triangular matrix and its transpose using subroutine CHLSKY, such that

$$\text{COV} = \text{L} * \text{L}^T .$$

This decomposition is used to weight the regression model.

Subroutine WIMTX calculates $\text{BWT} = \text{L}^{-1} * \text{B}$ and $\text{AWT} = \text{L}^{-1} * \text{A}$ by solving the lower triangular linear systems $\text{L} * \text{BWT} = \text{B}$ and $\text{L} * \text{AWT} = \text{A}$ for the vector BWT and the columns of AWT, respectively. The weighted model is then

$$\text{AWT} * \text{X} \approx \text{BWT} .$$

The matrix AWT is orthogonally decomposed using subroutine HDEC, such that

$$\text{AWT} = \text{Q} * \text{R}$$

where Q is $m \times m$ orthogonal, and R is $m \times n$ upper triangular.

Thus,

$$\begin{aligned} \text{AWT} * X &\approx \text{BWT} \\ (\text{Q} * \text{R}) * X &\approx \text{BWT} \\ \text{R} * X &\approx \text{Q}^T * \text{BWT} \end{aligned}$$

The first n rows of this overdetermined system form an $n \times n$ upper triangular system which is solved for the regression coefficients X using subroutine SOLVEH.

Subroutine HSITIM performs iterative improvement on the calculated regression coefficients.

SPACE REQUIRED

2527₈ = 1367₁₀

ACCURACY

The accuracy of RGRSN3 depends on the conditioning of the weighted data matrix and the error in the measurement of the random variables B .

TIMING

The timing is proportional to the quantity $N^3/6 + MN^2$. For $M = 10$ and $N = 6$, RGRSN3 takes 7 milliseconds on the CDC 7600.

PORTABILITY

RGRSN3 is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

REQUIRED RESIDENT ROUTINES

ULIBER, SQRTF

SUBROUTINE RGRSN4 (A,MA,M,N,B,ITS,TEMP,X,COV,MCOV,IBND,IER)

**DIMENSION OF
ARGUMENTS**

A(MA,N),B(M),TEMP(2*N*(M+1)+M*(K+5)),X(N),COV(MCOV,M)

LATEST REVISION

December 1973

PURPOSE

Given the regression model

$$A*x = b + \epsilon$$

where

A is an $m \times n$ known data matrix

b is a set of observations of m random variables, and

ϵ is an unknown vector of random errors which has a
known BANDED covariance matrix,

then RGRSN4

1. weights the regression model using the BANDED covariance matrix of ϵ , and
2. calculates the regression coefficients for the weighted model.

PURPOSE*(continued)*

The method used in RGRSN4 is theoretically the same as in RGRSN3. They differ in the numerical techniques used to weight the model. RGRSN4 takes advantage of the banded nature of the covariance matrix to decrease the number of operations necessary to do the weighting.

ACCESS CARDS

```
*FORTRAN,S=ULIB,N=RGRSN4
*COSY
```

USAGE

```
CALL RGRSN4 (A,MA,M,N,B,ITS,TEMP,X,COV,MCOV,IBND,IER)
```

ARGUMENTS**On Input****A**

Is an input two dimensional variable with row dimension MA and column dimension N. On input, A contains the data matrix for regression analysis.

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement in the calling program.

M

Is an integer input variable set equal to the actual number of rows of A used in the subroutine.

N

Is an integer input variable set equal to the column dimension of A.

B

Is a real input vector with dimension M. On input, B contains the elements of the right hand side.

ITS

Is an integer input variable set equal to the maximum number of iterations for iterative improvement. (ITS .GE. 1.)

TEMP

Is a real variable used internally for working storage. It must have dimension at least $2*N*(M+1)+M*(K+5)$.

COV

Is a real input two dimensional variable with row dimension MCOV and column dimension (IBND+1). On input, COV contains the non zero upper diagonals of the banded covariance matrix of the right hand side.

The format for the COV is as follows. The (i,j) element of the upper triangular part of the banded covariance matrix goes into $COV(j, j-(i-1))$, $i = 1, \dots, m$ and $j = i, \dots, \min(i+IBND, M)$. This is equivalent to placing the main diagonal in the first column of COV, and the first super diagonal into the second column of COV beginning with the second element. This process is repeated until the $IBND^{th}$ super diagonal is entered in column $IBND + 1$ of COV, beginning with element $IBND + 1$.

MCOV

Is an integer input variable set equal to the row dimension of COV as declared in the dimension statement of the calling program.

IBND

Is an integer input variable set equal to the number of non zero diagonals above the main diagonal of COV.

On Output

X

Contains the calculated regression coefficients for the weighted model.

On Output
(continued)

IER

Is an integer error flag.
= 1 if COV is singular
= 2 if the weighted data matrix is singular
= 0 otherwise.

ENTRY POINTS

RGRSN4, WTBNB, HDEC, SOLVEH, HSITIM, CHLBNB, BNDSL1, CBNDIM

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

If the input covariance matrix COV is singular, RGRSN4 prints the message

BANDED COVARIANCE MATRIX SINGULAR IN RGRSN4.

If the weighted data matrix is singular, RGRSN4 prints the message

WEIGHTED MATRIX SINGULAR IN RGRSN4.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized December 1973

ALGORITHM

The banded covariance matrix COV is decomposed into the product of a banded lower triangular matrix and its transpose using subroutine CHLBND, such that

$$\text{COV} = \text{L} * \text{L}^T$$

This decomposition is used to weight the regression model.

Subroutine WTBND calculates $\text{BWT} = \text{L}^{-1} * \text{B}$ and $\text{AWT} = \text{L}^{-1} * \text{A}$ by solving the banded lower triangular systems $\text{L} * \text{BWT} = \text{B}$ and $\text{L} * \text{AWT} = \text{A}$ for the vector BWT and the columns of AWT, respectively. The weighted model is then

$$\text{AWT} * \text{X} \approx \text{BWT}$$

The matrix AWT is orthogonally decomposed using subroutine HDEC, such that

$$\text{AWT} = \text{Q} * \text{R}$$

where Q is $m \times m$ orthogonal, and R is $m \times n$ upper triangular. Thus,

$$\begin{aligned} \text{AWT} * \text{X} &\approx \text{BWT} \\ (\text{Q} * \text{R}) * \text{X} &\approx \text{BWT} \\ \text{R} * \text{X} &\approx \text{Q}^T * \text{BWT} \end{aligned}$$

The first n rows of this overdetermined system form an $n \times n$ upper triangular system which is solved for the regression coefficients X using subroutine SOLVEH.

ALGORITHM
(continued)

Subroutine HSITIM performs iterative improvement on the calculated coefficients.

SPACE REQUIRED

2636₈ = 1438₁₀

ACCURACY

The accuracy of RGRSN4 depends on the conditioning of the weighted data matrix and the error in the measurement of the random variables B.

TIMING

The timing is proportional to the quantity

$$\frac{IBND^2 * M}{6} + MN^2$$

For M = 10, N = 6 and IBND = 2, RGRSN4 takes 4 milleseconds on the CDC 7600.

PORTABILITY

RGRSN4 is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

**REQUIRED RESIDENT
ROUTINES**

ULIBER, SQRTF

RGRSN5

SUBROUTINE RGRSN5 (A,MA,M,N,B,ITS,TEMP,X,BMX,MBMX,K,EPS,IER)

**DIMENSION OF
ARGUMENTS**

A(MA,N),B(M),TEMP(3*M*N+4*M),X(N),BMX(MBMX,K)

LATEST REVISION

December 1973

PURPOSE

Given:

A an $m \times n$ known data matrix, and

BMX an $m \times k$ matrix of repeated observations of m random variables,

- RGRSN5
1. calculates the row means and standard deviations of the matrix BMX,
 2. uses the standard deviations to weight the regression models, and
 3. calculates the regression coefficients of the weighted model.

ACCESS CARDS

*FORTRAN,S=ULIB,N=RGRSN5
*COSY

USAGE

CALL RGRSN5 (A,MA,M,N,B,ITS,TEMP,X,BMX,MBMX,K,EPS,IER)

ARGUMENTS

On Input

A

Is an input two dimensional variable with row dimension MA and column dimension N. On input, A contains the data matrix for regression analysis.

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the actual number of rows of A used in the subroutine.

N

Is an integer input variable set equal to the column dimension of A.

ITS

Is an integer input variable set equal to the maximum number of iterations for iterative improvements.
(ITS.GE.0)

TEMP

Is a real variable used internally for working storage. It must have dimension at least $3*M*N+4*M$.

BMX

Is a real input two dimensional variable with row dimension MBMX and column dimension K. On input, BMX contains the K observations of M random variables such that $BMX(I,J)$ is the J^{th} observation of the I^{th} random variable.

MBMX

Is an integer input variable set equal to the row dimension of BMX as declared in the dimension statement of the calling program.

K

Is an integer input variable set equal to the column dimension of BMX.

EPS

Is a real input variable set equal to the unit floating point machine precision (i.e., the smallest floating point number, such that $1. + EPS .GT. 1.$).

On Output

B

Contains the row means of the matrix BMX.

X

Contains the calculated regression coefficients for the weighted model.

IER

Is an integer error flag.

= 1 if $K \leq 1$.

= 2 if weighted data matrix is singular.

= 0 otherwise.

ENTRY POINTS

RGRSN5, WIVEC, MNSTD, HDEC, SOLVEH, HSITIM

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

If $K \leq 1$, RGRSN5 prints the message

***BMX NOT A MATRIX*.**

I/O
(continued)

If the weighted data matrix is singular, RGRSN5 prints the message

WEIGHTED MATRIX SINGULAR IN RGRSN5.

PRECISION

Single

REQUIRED ULIB
ROUTINES

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Standardized December 1973.

ALGORITHM

Subroutine MNSTD calculates the row means, B , and the row standard deviations, σ , for the matrix BMX . Let $D^{-1/2}$ be an $m \times m$ diagonal matrix with $D_{i,j} = 1/\sigma_i$, then RGRSN5 calculates the weighted model

$$AWT*x \approx BWT$$

where $AWT = D^{-1/2}*A$ and $BWT = D^{-1/2}*B$. The matrix AWT is orthogonally decomposed using subroutine HDEC, such that

$$AWT = Q*R$$

where Q is $m \times m$ orthogonal, and R is $m \times n$ upper triangular.

Thus,

$$\begin{aligned} \text{AWT}^*x &\approx \text{BWT} \\ (\text{Q}^*\text{R})^*x &\approx \text{BWT} \\ \text{R}^*x &\approx \text{Q}^{\text{T}}*\text{BWT} \end{aligned}$$

The first n rows of this overdetermined linear system form an $n \times n$ upper triangular system which is solved for the regression coefficients x using subroutine SOLVEH.

Subroutine HSITIM is used to perform iterative improvement on the calculated coefficients.

SPACE REQUIRED

2160₈ = 1136₁₀

ACCURACY

The accuracy depends on the conditioning of the weighted data matrix and the error in the measurement of the random variables.

TIMING

The timing is proportional to the quantity MN^2 .

For $M = 10$ and $N = 6$, RGRSN5 takes 2 milliseconds on the CDC 7600.

PORTABILITY

RGRSN5 is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

REQUIRED RESIDENT ROUTINES

ULIBER, SQRTF

SUBROUTINE RGRSN6 (A,MA,M,N,B,ITS,TEMP,X,BMX,MBMX,K,IER)**DIMENSION OF
ARGUMENTS**

A(MA,N),B(M),TEMP(2*N*(M+1)+M*(M+5)),X(N),BMX(MBMX,K)

LATEST REVISION

December 1973

PURPOSE

Given a known data matrix, A, and a matrix of repeated observations of the random variables, BMX, where the number of observations is greater than the number of variables, $K \geq M$,

then RGRSN6

1. calculates a sample covariance matrix for the random variables,
2. uses this estimated covariance matrix to weight the regression model, and
3. calculates the regression coefficients for the weighted model.

ACCESS CARDS

*FORTRAN,S=ULIB,N=RGRSN6
*COSY

USAGE

CALL RGRSN6 (A,MA,M,N,B,ITS,TEMP,X,BMX,MBMX,K,IER)

ARGUMENTS

On Input

A

Is an input two dimensional variable with row dimension MA and column dimension N. On input, A contains the data matrix for regression analysis.

MA

Is an integer input variable set equal to the row dimension of A as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the actual number of rows of A used in the subroutine.

N

Is an integer input variable set equal to the column dimension of A.

ITS

Is an integer input variable set equal to the maximum number of iterations for iterative improvement (ITS .GE. 0).

TEMP

Is a real variable used internally for working storage. It must have dimension at least $2*N*(M+1)+M*(M+5)$

BMX

Is a real input two dimensional variable with row dimension MBMX and column dimension K. On input, BMX contains the K observations of M random variables, such that, BMX(I,J) is the Jth observation of the Ith random variable.

MBMX

Is an integer input variable set equal to the row dimension of BMX as declared in the dimension statement of the calling program.

K
Is an integer input variable set equal to the column dimension of BMX, ($K \geq M$).

On Output

B
Contains the row means of the matrix BMX.

X
Contains the calculated regression coefficients for the weighted model.

IER
Is an integer error flag.
= 1 if $K \leq 1$.
= 2 if calculated covariance matrix is singular.
= 3 if the weighted data matrix is singular.
= 0 otherwise.

ENTRY POINTS

RGRSN6, COVARC, WIMIX, CHLSKY, CHSLV1, CHITM, HDEC, SOLVEH, HSITM

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

If $K \leq 1$, RGRSN6 prints the message

BMX NOT A MATRIX.

If the calculated covariance matrix is singular, RGRSN6 prints the message

COVARIANCE MATRIX SINGULAR IN RGRSN6.

I/O*(continued)*

If the weighted data matrix is singular, RGRSN6 prints the message

***WEIGHTED MATRIX SINGULAR IN RGRSN6*.**

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Standardized December 1973

ALGORITHM

The sample covariance matrix is calculated using subroutine COVARC. Then the inverse of the Cholesky decomposition of the sample covariance matrix is used to weight A and B, where B contains the row means of BMX. If $COV = L * L^T$ (the Cholesky decomposition), then

$$AWT = L^{-1} * A \text{ and } BWT = L^{-1} * B$$

These are calculated in subroutine WIMTX. The weighted model is then

$$AWT * X \approx BWT$$

FUNCTION RLHPTS (AWT,X,BWT,C,D,TEMP,MAWT,M,MC,K,N,ISW)

DIMENSION OF ARGUMENTS AWT(MAWT,N),X(N),BWT(M),C(MC,N),D(K),TEMP(M*(2*N-K+4)+N*(K+4)+K)

LATEST REVISION December 1973

PURPOSE RLHPTS calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable.

ACCESS CARDS *FORTRAN,S=ULIB,N=RLHPTS
 *COSY

USAGE PROB = RLHPTS (AWT,X,BWT,C,D,TEMP,MAWT,M,MC,K,N,ISW)

ARGUMENTS

On Input AWT
 Contains the data matrix for the regression model.

On Input
(continued)

X

Contains the calculated regression coefficients.

BWT

Contains the vector of observations of the random variables in the regression model.

C

Contains the matrix elements of the linear hypotheses, where a row of C corresponds to a single hypothesis.

TEMP

Is a vector used internally for working storage. It must have dimension at least $M*(2*N-K+4)+4*(K+4)+K$.

MAWT

Is an integer input variable set equal to the row dimension of the matrix AWT as declared in the dimension statement of the calling program.

M

Is an integer input variable set equal to the row dimension of AWT actually used in the function.

MC

Is an integer input variable set equal to the row dimension of the matrix C as declared in the dimension statement of the calling program.

K

Is an integer input variable set equal to the row dimension of C actually used in the function.

N

Is an integer input variable set equal to the column dimension of A.

On Output

ISW

Is an integer variable used as an error flag.

= 1 if the set of hypotheses are inconsistent.

= 0 otherwise.

ENTRY POINTS

RLHPTS, CNSLSQ, CNLSIM, HOUSML, SOLVEH, SOLVEL, HDEC, HDEC2,
FINTGL, GAUSS

COMMON BLOCKS

None

I/O

All messages are printed using ULIBER.

If the set of hypotheses are inconsistent, the function
prints the message

HYPOTHESES INCONSISTENT IN FUNCTION RLHPTS.

If K = 0, the function prints the message

NO HYPOTHESES IN RLHPTS.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Standardized December 1973

ALGORITHM

The quantity $\|AWT*\hat{x} - BWT\|_2^2$ is calculated and stored in SSR, where \hat{x} contains the regression coefficients.

The quantity $\|AWT*\tilde{x} - BWT\|_2^2$ is calculated and stored in SSC, where \tilde{x} is the solution vector for the constrained least squares problem, $\min_x \|AWT*x - BWT\|_2$ subject to the equality constraints $C*x = D$. The F-statistic

$$FSTAT = ((SSC-SSR)/K)/(SSR/(M-N))$$

is calculated. This is the test statistic for the hypotheses $C*x = D$.

Finally, the function value RLHPTS is

$$RLHPTS = 1. - \int_0^{FSTAT} h(F)dF$$

where $h(F)$ is the F-density function.

SPACE REQUIRED

$$5051_8 = 2601_{10}$$

ACCURACY

The accuracy depends on the conditioning of the matrix AWT and the error in the measurement of the random variables BWT.

TIMING

The timing is proportional to $NK^2 + M(N-K)^2$. For $N = 5$, $M = 10$ and $K = 1$, RLHPTS takes 2.5 milliseconds on the CDC 7600.

PORTABILITY

RLHPTS is written in Standard FORTRAN. It has been tested on the CDC 6600 and 7600.

**REQUIRED RESIDENT
ROUTINES**

SQRTF, ATANF, ULIBER, RBAREX

SUBROUTINE RNDEV (X1,X2,RANF)**LATEST REVISION** December 1, 1973**PURPOSE** RNDEV produces two independent random deviates X1 and X2 each from the normal distribution with mean 0 and variance 1.**ACCESS CARDS** *FORTRAN,S=ULIB,N=RNDEV
 *COSY**USAGE** CALL RNDEV (X1,X2,RANF)**ARGUMENTS****On Input** X1, X2
 Are ignored.

RANF

An external function provided by the user to calculate uniform random deviates in the interval (0,1). RANF must be declared external in the calling subroutine.

9.RNDEV.2

On Output

X1,X2

Two independent normal random deviates with mean 0 and variance 1.

ENTRY POINTS

RNDEV

SPECIAL CONDITIONS

None

COMMON BLOCKS

None

I/O

None

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

**USER-PROVIDED
ROUTINES**

Function RANF(X)

ARGUMENTS FOR RANF

**On Input
for RANF**

X

A dummy (used only because FORTRAN requires a function to have an argument).

**On Output
for RANF** If $Y = \text{RANF}(X)$, then Y is a uniform random deviate between 0 and 1.

SPECIALIST Jack Miller, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY This appeared as A.C.M. Algorithm 267.

ALGORITHM The method is given by Box, G.E.P. and Muller, M.E., 1958: A note on the generation of normal random deviates. *Ann. Math. Stat.* 29, 610-611.

SPACE REQUIRED 40₈

ACCURACY Given a source of uniform random deviates, the Box-Muller algorithm is known to produce independent normal random deviates with mean 0 and variance 1.

TIMING Dependent on the timing required by the user-supplied RANF. Using system RANF \sim .166 sec/call.

PORTABILITY RNDEV is written in standard FORTRAN. It has been tested on the CDC 6600 and 7600.

**REQUIRED RESIDENT
ROUTINES** SIN, COS, LOG

SUBROUTINE SPAL (T,N,P,DUM,NA,IDT,SR,FF)**LATEST REVISION**

August 1968

**DIMENSION OF
ARGUMENTS**

T(MT),P(MT/2),DUM(MT/2)

Where $MT = 2^{**}K+4$, and K is the smallest integer such that $2^{**}K \geq N$, (i.e., T must be dimensioned at least 4 more than the smallest power of 2 which will include all points in the time series).

PURPOSE

SPAL computes power spectral estimates for a one-dimensional stationary time series. SPAL provides several options for detrending the time series data. This is controlled through the value of the input parameter IDT.

ACCESS CARDS

*FORTRAN,S=**ULIB**,N=SPAL
*COSY

USAGE

CALL SPAL (T,N,P,DUM,NA,IDT,SR,FF)

ARGUMENTS

On Input

T

A real vector variable with dimension MT. (See 'DIMENSION OF ARGUMENTS'.) T contains the time series data which is assumed to be at constant time intervals. The time series must include at least 8 points.

N

An integer input variable set equal to the actual number of data points in the time series.

DUM

A real array with dimension at least MT/2. (See 'DIMENSION OF ARGUMENTS'.) It is a scratch array used internally for working storage.

NA

An integer input variable set equal to the number of frequency values to be averaged in calculating the spectral estimates. If NA = 0, then the value of 5 is assumed. This is approximately equivalent to an autocorrelation lag of 20%.

IDT

An input integer flag which indicates the type of detrending of the time series data to be performed.

- = 0 Causes no detrending.
- = 1 Causes removal of the mean.
- = 2 Causes linear detrending.
- = 3 Causes quadratic detrending.

SR

The "sample rate" for the input time series; that is, the constant time interval between points in the time series.

On Output P
A real vector array with dimension at least $MT/2$. On output, P contains the power spectral estimates.

FF
A real output variable which contains the frequency spacing for the output spectral estimates.

ENTRY POINTS SPAL, FFT (Los Alamos version), RFFT, QUAD, TAPER, SMSPECT

COMMON BLOCKS None

I/O None

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Robert Lackman, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY SPAL is a subroutine from the Los Alamos Scientific Laboratory, Los Alamos, New Mexico. Documentation for NSSL written by Jo Walsh, NCAR, Boulder, Colorado 80303.

ALGORITHM

The subroutine proceeds as follows:

The time series is detrended. A curve (corresponding to the value of the input parameter IDT) is fit to the data. This trend is subtracted from the data, leaving residuals for the analysis.

The time series is tapered at both ends. A rising half of a cosine bell is used for the front ten percent of the series, and a falling half of a cosine bell is used for the last ten percent of the series. This step is needed to reduce side lobes of the spectral window.

If the number of elements in the series is not a power of 2, zero values are added in equal numbers to both the front and the end of the series to make the number of values a power of 2.

All Fourier coefficient pairs are computed by means of the fast Fourier transform.

The power, $(a_j^2 + b_j^2)$, for all Fourier coefficients is computed, and a running average of NA power values is computed.

The resulting running average is stored in P as the final spectral estimates. Any sequence of the spectral estimates which are NA values apart may be used for significance or other tests.

SPAL does not use the method of lagged autocorrelation values to estimate the power spectrum. Since the advent of the Fast Fourier transform, spectral estimates can be provided more rapidly by computing all Fourier coefficients.

SPACE REQUIRED $3151_8 = 1641_{10}$ **TIMING**

The timing is proportional to the quantity
 $1.8 \cdot 10^{-5} \cdot N \cdot \text{LOG}_2(N)$ seconds.

PORTABILITY

SPAL was written on the CDC 6600 and run on both CDC 6600
and CDC 7600.

**REQUIRED RESIDENT
ROUTINES**

LOGF, SIN, COS, SQRT, IBAIEX

SUBROUTINE SPECFT (XIN,YIN,T,X,Y,F,NUM,NAMEX,NAMEY,TSCALE)**LATEST REVISION**

December 21, 1973

**DIMENSION OF
ARGUMENTS**

XIN(NUM),YIN(NUM),T(NUM),X(NUM),Y(NUM),F(NUM)

PURPOSE

To compute real power spectra and provide log-log plots of

- A single power spectrum of a time series input
- A cross spectrum of 2 time series inputs including

Power spectrum of series A

Power spectrum of series B

Cospectrum of A and B

Quadrature of A and B

Coherence of A and B

Phase of A and B

The choice of either option is controlled by parameter ISPEC.
See inputs below for common block SPECK1.

ACCESS CARDS

*FORTRAN, S=ULIB, N*SPECFT
*COSY

9.SPECFT.2

USAGE

CALL SPECFT (XIN,YIN,T,X,Y,F,NUM,NAMEX,NAMEY)

ARGUMENTS

On Input

XIN

Input time series data of constant DT, dimensioned by NUM.

YIN

Equal to XIN if ISPEC = 1 (default value of ISPEC), or a second time series if ISPEC. \neq 1.

T

The sampling times of data arrays XIN and YIN. Dimension size is NUM.

X, Y, F

Work arrays dimensioned by NUM.

If XIN, YIN and T may be destroyed, let X = XIN, Y = YIN, F = T.

NUM

The number of points in arrays XIN, YIN, T, X, Y and F.

NAMEX

A 20 Hollerith name of data series XIN and its units, e.g., NAMEX(1) = 10HVERTICAL V, NAMEX(2) = 10HEL M/S.

NAMEY

A 20 Hollerith name of data series YIN and its units.

TSCALE

A scale factor which would convert the times in array T to seconds. If T is in minutes, TSCALE = 60, etc.

SPECIAL CONDITIONS

There are a number of options available to the user. These are described under the section "COMMON BLOCKS, SPEC1". Also, there are options on the dimensioning of the calling argument list. These are described under "SPACE REQUIRED".

COMMON BLOCKS**Optional Input**

COMMON/SPEC1/ISG,MSUM,ISPEC,ILIMS,CONF,LINTR,PERTP,PERZR,IPRNT,LAB(6),NPLOTS,IEND,NOTEPR

The list of parameters in "COMMON BLOCK SPEC1" may be set at execution time in the user's driver routine. Each item is defined below and its default value is given.

ISG

Provides a choice of reliability enhancement of the raw spectrum. Modes can be summed into wider frequency bands, or the user can select one of 3 running windows to smooth the spectrum. The choices are listed below.

- = 0 Sets of adjacent raw spectral modes are summed into MSUM frequency bands of width $DF \cdot NH / MSUM$ where DF is the elementary frequency band, NH is the number of raw modes, and $MSUM$ is the number of desired sums.
- = 1,2,or 3 The spectral estimates are smoothed by a running window of $MSUM$ points width.
- = 1 An $MSUM$ point running average.
- = 2 $MSUM$ triangular weights (Bartlett)
- = 3 $MSUM$ weights from a hamming window.

See Ref. (1) for a discussion of these smoothing windows and their associated degrees of freedom. Since a true centered running smoother is not possible for $MSUM/2$ points at each of the spectrum ends, two alternate smoothing techniques are available for them which are described below under item IEND. Moreover, since degree of freedom calculations based upon the full running window do not apply at the ends of the spectrum, confidence bands, as described under item ILIMS, are not included over these portions of the spectral plots.

Note: MSUM has a different meaning based upon the choice of ISG. (The default value of ISG is 3.)

Optional Input
(continued)

MSUM

If ISG = 0, MSUM is the number of spectral estimates obtained by summing adjacent raw spectral modes into frequency bands. MSUM on the order of NUM/100 is recommended.

If ISG = 1,2,or 3, MSUM is the number of non zero weights in the smoothing window of the raw spectra. A recommended value of this MSUM would be on the order of 25 to 51. The resulting number of degrees of freedom for the spectral estimates is computed internally for either ISG choice and is displayed on the spectral plots. The choice of MSUM is a tradeoff between spectral reliability and resolution and may require some experimentation on the user's part.

MSUM must be odd for ISG \neq 0 and will be set to $MSUM = 2*(MSUM/2) + 1$ if it is input as an even value. The default value of MSUM is NUM/100.

ISPEC

- = 1 Tells SPECFT that only a single power spectrum of XIN is to be computed and displayed.
- \neq 1 Requests dual power spectra of the XIN and YIN data series along with the other cross spectrum parameters. All such items are displayed on plots.

The default value of ISPEC is 1.

ILIMS

Controls an option for applying dashed line confidence bounds to the power spectra. The listed references indicate that these confidence bands only apply to flat spectra from a time series of Gaussian random variables.

- = 1 Confidence bands are added to the spectral plots.
- \neq 1 No confidence bands are added.

See CONF definition below.

The default value of ILIMS is 0.

CONF

If ILIMS = 1, CONF is the percentage confidence of the dashed line bands to be added to the spectral plots, such as 90, or 99, percent.

The default value of CONF is 99.

LINTR

- = 1 The mean and linear trend will be removed from XIN and YIN before transforming.
- ≠ 1 The mean and linear trend are not removed.

The default value of LINTR is 1.

Note: If the user inputs LINTR ≠ 1, he must supply a reasonable alternative, or be prepared to interpret his results accordingly.

PERTP

The 2 ends of each input data series are bell tapered to zero to imply data periodicity. PERTP is the percentage of the total number of data points which is tapered at each end. If PERTP = 5, 5 percent of the XIN and YIN data series are bell tapered at each end.

The default value of PERTP is 5.

PERZR

Some users prefer to extend their XIN and YIN data by adding zeros. PERZR is the percentage of zeros to be added in relation to the original NUM data points. If NUM = 4000 and PERZR = 10, approximately 400 zeros would be added to XIN before transforming. Make sure that the XIN, YIN, X, Y, T, and F arrays are dimensioned large enough to hold any added zeros.

The default value of PERZR is 0.

IPRNT

- = 0 No output print.
- = 1 Printing of data series statistics, etc.
- = 2 Same as 1 plus a print of the raw and smoothed spectra as well as all cross spectral parameters. This option is discouraged when a large number of points are transformed because of the huge amount of print which will be generated.

The default value of IPRNT is 0.

LAB(6)

A 60 character Hollerith main title placed at the top of all SPECFT plots.

The default value of LAB(6) is blanks.

Optional Input
(continued)

NPLOTS

The output plots, listed as 1 through 6 under header "I/O", can be single plots of all points or can be split into a number, NPLOTS, of plots all having an equal number of points. This option is useful when the number of points to transform is large or when expanded plot resolution is desirable.

The default value of NPLOTS is 1.

Note: If a set of plots is also wanted for the other cross spectral parameters, one need only delete the instruction NPLOTS = 1 in subroutine SPECFT at approximately instruction number 799.

In order to delete all plots of the input data and still retain 1 plot of each spectral parameter use NPLOTS = 0.

IEND

The full application of a centered running window for smoothing the raw spectrum is not possible for MSUM/2 points at each end of the spectrum when the window contains MSUM weights, two alternate techniques are available in subroutine AVERG for handling these end points. They are:

= 1 Apply a smoothing window of a reduced number of weights. 1 weight for point 1, 3 for point 2, ...MSUM-1 weights for point MSUM/2.

Note: This method yields first and last points that are raw modes and as such give incorrect values of coherence = 1 and phase = 0.

= 0 Effectively extend the ends of the raw spectra by reflection of the modes about the first and last values ignoring the DC or ZEROth mode.

The default value of IEND is 0.

NOTEPR

Because the MSUM/2 points at each end of the power spectrum are not true centered running windows when the ISG \neq 0 option is used, a warning note to this effect can be printed on each spectral plot.

= 0 The warning note is written.

= 1 The warning note is not written.

The default value of NOTEPR is 0.

Internal

```
COMMON/SPEC2/NSUM,NDEG,FNYQ,DT,LV(6,12),NIN,ALPHA,ALPHB,
TCON,BCON
```

```
COMMON/SPEC3/ICOH,IPH,LABSIZ,ICONL,ICONF
```

```
COMMON/SPEC4/LDIM,W(LDIM),SB(LDIM),SE(LDIM)
```

All items in SPEC2 and SPEC3 are internally computed.

SPEC4 parameters involve the application of the spectra smoothing windows. LDIM, the dimension size of W, SB, and SE is set in a SPECFT data statement. The smoothing weights, W, are internally computed. SB and SE are storage arrays for raw modes during the smoothing process. If the user wishes to use more weights than the presently set LDIM he should change LDIM in its data statement and insert the resulting /SPEC4/ COMMON BLOCK in subroutines SPECFT and AVERG.

I/O

Input is via "ARGUMENTS" and "COMMON BLOCK SPEC1". Output is microfilm plots of

- (1) XIN
- (2) XIN after linear trend and mean removal, and tapering.
- (3) YIN
- (4) YIN after linear trend and mean removal, and tapering.
- (5) FX Power spectrum of XIN.
- (6) FY Power spectrum of YIN.
- (7) CRXY Cospectrum of XIN and YIN.
- (8) QXY Quadrature of XIN and YIN.
- (9) C Coherence of XIN and YIN.
- (10) P Phase relation of XIN and YIN.

If ISPEC = 1, only plots (1), (2), and (5) are generated. Additional paper print is provided based upon the selection of input value IPRNT as described above.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

INTRINSIC ROUTINES

IDIOT replaces the ULIB IDIOT routine. This IDIOT package is a slight modification of the ULIB version; however, in terms of I/O, as seen by the user, it is identical. All calls to the IDIOT routine within the user's program will utilize this package, not the standard IDIOT available on ULIB. There should be no problem with its use.

**ADDITIONAL INTERNAL
SUBROUTINES**

NZERO

The requested NUM data points in XIN cannot be transformed by REALFT if NUM when factored contains a prime number larger than 19. Thus, NUM is truncated to an even number and reduced by 2 until its factored primes are all less than 19. This number of points \leq NUM is then used in SPECFT in place of NUM.

Note: In general, very few points need be deleted before the condition is satisfied.

REALFT

This Fast Fourier routine is a real, 1 dimensional Fourier transform which is significantly faster than the general FOURT ULIB routine. REALFT is not constrained to powers of 2 on input, but can transform a number of points which when factored into prime numbers has no prime greater than 19. This restriction need not concern the user, since SPECFT internally reduces the user requested NUM until it meets this condition and the actual number of points transformed is printed on the spectral plots.

REALFT accepts an equally spaced data series and returns the real and imaginary Fourier coefficients.

TAPER

Provides a bell taper to PERTP percent of both ends of the XIN and YIN data series.

SUMODE

This subroutine sums adjacent modes into spectral estimates when ISG = 0.

AVERG

This subroutine smooths the raw spectra by one of 3 window choices: running average, triangular weights, or hamming weights.

Special treatment is required for the MSUM/2 points at each end of the spectrum if the running window is over MSUM points. See the definitions of "COMMON BLOCK SPEC1" inputs ISG, IEND, and NOTEPR.

RLT

Removes the mean and linear trend from the XIN and YIN time series when LINTR = 1.

CONLIM

Performs an integration of the chi-squared distribution to obtain the upper and lower confidence bounds for the power spectral estimates. These bounds are dependent upon the confidence requested and the number of degrees of freedom the spectral estimates have. NDEGF is computed in SPECFT according to formulas provided by the references therein cited.

LBINIT

This subroutine sets up all plotting labels.

TMUNIT

This subroutine determines the time scale to be used in the plotting of the input data series.

PRFX

Used to print the spectral modes.

PTITLE

Prints several items on the spectral plots.

ORIGINATOR IDIOT is a ULIB routine. REALFT was developed by Gordon Sande, University of Chicago. All other routines were developed by Robert L. Lackman, NCAR, Boulder, Colorado 80303.

SPECIALIST R. L. Lackman, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Subroutine CORFFT, now called SPECFT, was originally written for ULIB in March 1972 and revised December 1973.

Method An input number of points, NUM, is specified in the SPECFT argument list. This, however, might not be the actual number of transformed points. If a certain percentage of zeros are requested at the end of the time series, the number of points will be a larger number. If no zeros are added, the actual number of points transformed, NIN, might be less than the input NUM. This occurs because the FFT subroutine used, REALFT, cannot transform a number which has a prime factor larger than 19. Thus SPECFT will take the input number, NUM, make it even and reduce it in steps of 2 until it becomes a number having prime factors all of 19 or less. Generally this occurs within only a few points.

The Fast Fourier routine REALFT is used to transform data from an equally spaced time series. The imaginary and real coefficients which are returned are combined as follows to provide the cross spectrum:

If XR(I) and XI(I) are the real and imaginary coefficients of the Ith mode for time series X, then

$$\begin{aligned} \text{X Spectrum} = \text{FX(I)} &= \text{XR(I)*XR(I)} + \text{XI(I)*XI(I)} \\ \text{Y Spectrum} = \text{FY(I)} &= \text{YR(I)*YR(I)} + \text{YI(I)*YI(I)} \\ \text{Cospectrum} = \text{CS(I)} &= \text{XR(I)*YR(I)} + \text{XI(I)*YI(I)} \\ \text{Quadrature} = \text{QD(I)} &= \text{YR(I)*XI(I)} - \text{XR(I)*YI(I)} \end{aligned}$$

Then as requested by input parameter ISG, defined above under "COMMON BLOCK SPEC1" inputs, these 4 items are summed into larger frequency bands or subjected to a smoothing window to enhance their reliability. The resulting FX(J), FY(J), CS(J), and QD(J) are then used to compute the coherence and phase relationships via

$$\begin{aligned} \text{C(J)} &= (\text{CS(J)*CS(J)} + \text{QD(J)*QD(J)})/(\text{FX(J)*FY(J)}) \\ \text{P(J)} &= \text{ATAN2(QD(J),CS(J))*DEGREES} \end{aligned}$$

The spectral estimates are normalized to

$$(\text{TIME SERIES UNITS})^{**2}/\text{HZ}$$

by dividing each FX(J) and FY(J) by the NYQUIST frequency in HZ.

In addition, the spectra are made one sided by putting all power at + frequencies. See Blackman & Tukey, *The Measurement of Power Spectra*, p. 86.

Each spectral mode after normalization is considered to be the power in a frequency interval of $DF = FNYQ/NH$, except for modes 1 and $NH + 1$. They are over $DF/2$, where FNYQ is the NYQUIST frequency and NH is 1/2 the number of transformed points. Thus, total power = data series variance when the mean has been removed =

$$(\text{X(1)} + \text{X(NH + 1)}) * DF/2 + \text{THE SUM (I = 2, NH) OF X(I) * DF}$$

When ISG = 0 and raw modes are summed into wider frequency bands, mode 1 (DC power) is not included and the $NH + 1$ mode is multiplied by .5 to account for its $DF/2$ frequency width.

Method
(continued)

The DC power, Mode 1, is printed on the spectral plots. It is not identically zero because the TAPER operation after mean removal introduces a slight non zero mean.

If the XIN data series is Gaussian with a flat spectrum over the smoothing window, the spectral estimates will be distributed as a chi-square with K degrees of freedom. See Ref. (3), page 80. If ILIMS = 1, the chi-square distribution is integrated in subroutine CONLIM to give upper and lower bounds for a selected confidence level (see CONF). On pages 19-25 of Ref. (1), it is shown that the number of degrees of freedom for the spectral estimates of a smooth spectrum is $K = WE/DF$, where WE is the equivalent frequency width of the spectral smoother and DF is the elementary frequency band. For the 3 smoothers, running ave, triangular weights, and hamming, $WE = 1.0*W, .75*W$, and $.63*W$ where W is the window width at the zero amplitude line. For our purposes $W = MSUM*DF$, thus the degrees of freedom would be $K = MSUM, .75*MSUM, .63*MSUM$, respectively for a smooth spectrum. $K = NH/MSUM$, the number of summed modes, is the number of degrees of freedom associated with the alternate summing technique.

The number of degrees of freedom is substantially less if the spectrum is not smooth. Page 24 of Ref. (1) gives an approximation formula for K which illustrates this point. It is

$$K = (P_1 + P_2 + \dots)^2 / (P_1^2 + P_2^2 + \dots)$$

For $P_1 = P_2 = P_3 = \dots = P_{MSUM}$, K will = $MSUM$, but for P_j much greater than any other P , K approaches 1. Thus, the flat spectrum bands are probably somewhat optimistic. The confidence bands are not included at the ends of the spectra which are smoothed by running windows, since some artificial

adjustment is required there. (See the definition of IEND.)
 Moreover, confidence bands are only provided when there are
 4 or more degrees of freedom.

REFERENCES

- (1) Blackman & Tukey, 1958: *The Measurement of Power Spectra*. Dover Publications Inc., New York, N.Y.
- (2) Cooley, Lewis, and Welch, 1967: *The FFT Algorithm and Its Applications*. IBM Research Document No. RC1743.
- (3) Jenkins & Watts, 1968: *Spectral Analysis ...*, Holden-Day Publishing Co.

SPACE REQUIRED

SPECFT size is approximately 25,000 cells plus the array sizes of XIN, YIN, T, X, Y, and F, as dimensioned in the calling driver. The number of actual needed arrays will thus vary from 2 to 6 depending on the user's needs. The call for a single spectrum without array save would be

```
CALL SPECFT (XIN,XIN,T,XIN,XIN,T,NUM,NAMEX,NAMEX,TSCALE).
```

A cross spectrum with array save would be

```
CALL SPECFT (XIN,YIN,T,X,Y,F,NUM,NAMEX,NAMEY,TSCALE) .
```

A maximum decimal value of NUM, the number of points that can be transformed, is thus approximately 40000 / (THE NUMBER OF NEEDED ARRAYS) .

ACCURACY

The chi-squared integration is accurate to approximately 4 significant digits. This integration is used to establish confidence bands for the spectra. The running mode averages in subroutine AVERG are individually computed in order to avoid round off error accumulation.

TIMING

A 19890 point single spectrum with NPLOTS = 0 took 34 seconds on the CDC 7600.

PORTABILITY

Discouraged because of the use of title encoding.

**REQUIRED RESIDENT
ROUTINES**

OUTPTC, OUTPTS, ATANF, ATAN2, COSF, EXPF, LOGF, SQRT, KODER, Q8QERR, Q8QRSD, RBAIEX, RBAREX, IBAIEX, ACGOER and the DD80 plot package.

NCAR Software Support Library Volume 3

Editors: Jeanne C. Adams
Alan K. Cline
Margaret A. Drake
Roland A. Sweet

(10)

INPUT/OUTPUT



(11)

UTILITY ROUTINES



(12)

GRAPHICS



(13)

UTILITY PROCESSORS



CONTENTS
VOLUME III

[CHAPTER 1]	SOLUTION OF NON-LINEAR SYSTEMS	
	DETERMINATION OF ROOTS OF POLYNOMIALS	
	DBDNZRO	1.DBDNZRO.1
	DCPOLY	1.DCPOLY.1
	DRPOLY	1.DRPOLY.1
	RTNI	1.RTNI.1
[CHAPTER 2]	INTERPOLATION, APPROXIMATION AND SMOOTHING	
	BSL1NT	2.BSL1NT.1
	BSL2NT	2.BSL2NT.1
	CUBSPL	2.CUBSPL.1
	CURV	2.CURV.1
	HRM1NT	2.HRM1NT.1
	HRM2NT	2.HRM2NT.1
	KURV	2.KURV.1
	KURVP	2.KURVP.1
	QURV	2.QURV.1
	SPLPAK	2.SPLPAK.1
	SURF	2.SURF.1
	TRIANGLE	2.TRIANGLE.1
[CHAPTER 3]	SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/ EIGENVECTOR ANALYSIS	
	BDSLV	3.BDSLV.1
	BND3	3.BND3.1
	CHLSLV	3.CHLSLV.1
	EIGCFA	3.EIGCFA.1
	EIGHFS	3.EIGHFS.1
	EIGRFA	3.EIGRFA.1
	EIGSFM	3.EIGSFM.1
	EIGSFS	3.EIGSFS.1
	EIGSTM	3.EIGSTM.1
	EIGSTS	3.EIGSTS.1
	HSHSLV	3.HSHSLV.1
	INVMTX	3.INVMTX.1
	LINEQSV	3.LINEQSV.1
	SUPRLS	3.SUPRLS.1
	SVDSLV	3.SVDSLV.1
	TRDI	3.TRDI.1
	TRDIP	3.TRDIP.1

[CHAPTER 4]	NUMERICAL INTEGRATION (QUADRATURE)	
	ADQUAD	4.ADQUAD.1
	GAUSS	4.GAUSS.1
	SIMPSN	4.SIMPSN.1
[CHAPTER 5]	SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS	
	AM1	5.AM1.1
	DIFSUB	5.DIFSUB.1
	GEAR	5.GEAR.1
	RK1	5.RK1.1
[CHAPTER 6]	SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS	
	BLKTRI	6.BLKTRI.1
	POIS	6.POIS.1
	PWSCRT	6.TWSCRT.1
	PWSCSP	6.PWSCSP.1
	PWSCYL	6.PWSCYL.1
	PWSSSP	6.PWSSSP.1
[CHAPTER 7]	EVALUATION OF SPECIAL MATHEMATICAL FUNCTIONS	
	BESLIK	7.BESLIK.1
	BESLJY	7.BESLJY.1
	CXERFC	7.CXERFC.1
	ELIPE	7.ELIPE.1
	ELIPK	7.ELIPK.1
	EXPINT	7.EXPINT.1
	HYPER	7.HYPER.1
[CHAPTER 8]	FAST FOURIER ANALYSIS	
	FFT	8.FFT.1
	FFTPOW2	8.FFTPOW2.1
[CHAPTER 9]	STATISTICAL ANALYSIS AND RANDOM NUMBER GENERATION	
	CNFPRB	9.CNFPRB.1
	RGRSN1	9.RGRSN1.1
	RGRSN2	9.RGRSN2.1
	RGRSN3	9.RGRSN3.1
	RGRSN4	9.RGRSN4.1
	RGRSN5	9.RGRSN5.1
	RGRSN6	9.RGRSN6.1
	RLHPTS	9.RLHPTS.1
	RNDEV	9.RNDEV.1
	SPAL	9.SPAL.1
	SPECFT	9.SPECFT.1

CHAPTER 10

SPECIAL-PURPOSE INPUT/OUTPUT ROUTINES

BRANRD	10.BRANRD.1
CORFOR	10.CORFOR.1
READLX	10.READLX.1
TAPECY	10.RAPECY.1
UBLOK	10.UBLOK.1
UZBLOK	10.UZBLOK.1

CHAPTER 11

DATA PROCESSING UTILITY ROUTINES

BSEARCH	11.BSEARCH.1
CHCONV	11.CHCONV.1
CONV360	11.CONV360.1
DATE	11.DATE.1
HOURS	11.HOURS.1
LCKSUM	11.LCKSUM.1
MOVE	11.MOVE.1
UCHAR	11.UCHAR.1

CHAPTER 12

COMPUTER GRAPHICS

INTRODUCTION

AUTOGRAPH	12.AUTOGRAPH.1
CONREC Standard	12.CONREC STANDARD.1
CONRECQCK	12.CONRECQCK.1
CONRECSMTH	12.CONRECSMTH.1
DASHCHAR	12.DASHCHAR.1
DASHLINE	12.DASHLINE.1
DASHSMTH	12.DASHSMTH.1
HAFTON	12.HAFTON.1
ISOSRF	12.ISOSRF.1
ISOSRFHR	12.ISOSRFHR.1
PWRX	12.PWRX.1
PWRY	12.PWRY.1
PWRZ	12.PWRZ.1
SCROLL	12.SCROLL.1
SRFACE	12.SRFACE.1
SUPMAP	12.SUPMAP.1
VELVEC	12.VELVEC.1

APPENDIX 1: PROCESSING ARRAYS

APPENDIX 2: TRANSFORMATIONS

CHAPTER 13

FILE MANIPULATION, TEXT EDITING, PROGRAM
PREPROCESSING AND DEBUGGING

EDITOR	13.EDITOR.1
FIDEL	13.FIDEL.1
FLEX	13.FLEX.1
FRED	13.FRED.1

DESCRIPTION OF SUBROUTINES**VOLUME III****[CHAPTER 1]****SOLUTION OF NON-LINEAR SYSTEMS
DETERMINATION OF ROOTS OF POLYNOMIALS**

- DBNDZRO** Computes a posteriori error bounds and improvements to roots of polynomials.
- DCPOLY** Finds all the roots of a polynomial with complex coefficients using double precision.
- DRPOLY** Finds all the roots of a polynomial with real coefficients using double precision.
- RTNI** Finds a root of a given non-linear equation.

[CHAPTER 2]**INTERPOLATION, APPROXIMATION AND SMOOTHING**

- BSL1NT** Performs one-dimensional Bessel interpolation of selected order for values and derivatives.
- BSL2NT** Performs two-dimensional Bessel interpolation of selected order for values and derivatives.
- CUBSPL** Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives.
- CURV** Performs one-dimensional interpolation, differentiation, and integration using splines under tension.

[CHAPTER 2]
(continued)

INTERPOLATION, APPROXIMATION AND SMOOTHING (continued)

HRMINT	Performs one-dimensional Hermite interpolation of selected order for values and derivatives.
HRM2NT	Performs two-dimensional Hermite interpolation of selected order for values and derivatives.
KURV	Performs interpolation of a parameterized curve in the plane using splines under tension.
KURVP	Performs interpolation of a parameterized closed curve in the plane using splines under tension.
QURV	Performs interpolation of a parameterized curve in space using splines under tension.
SPLPAK	Performs least squares fitting of multidimensional cubic splines to arbitrarily located data.
SURF	Performs two-dimensional interpolation using a bi-spline under tension.
TRIANGLE	Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane.

[CHAPTER 3]

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS

BDSLV	Solves a banded linear system.
BND3	Solves a tridiagonal linear system using Gaussian elimination with partial pivoting.
CHLSLV	Solves a real, symmetric, positive definite linear system by Cholesky decomposition.

[CHAPTER 3]
(continued)

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS
(continued)

EIGCFA	Computes all the eigenvalues and selected eigenvectors of a general complex matrix.
EIGHFS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix.
EIGRFA	Computes all the eigenvalues and selected eigenvectors of a general real matrix.
EIGSFM	Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric matrix.
EIGSFS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix.
EIGSTM	Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix.
EIGSTS	Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix.
HSHSLV	Solves a real overdetermined linear system using Householder transformations.
INVTX	Computes the inverse of a general real matrix using Gaussian elimination with full pivoting.
LINEQSV	Solves a real linear system using Gaussian elimination with partial pivoting.

[CHAPTER 3]
(continued)

SOLUTION OF LINEAR SYSTEMS AND EIGENVALUE/EIGENVECTOR ANALYSIS
(continued)

- SUPRLS** Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.
- SVDSL** Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.
- TRDI** Solves a diagonally dominant tridiagonal linear system using Gaussian elimination without pivoting.
- TRDIP** Solves a diagonally dominant periodic tridiagonal linear system using Gaussian elimination without pivoting.

[CHAPTER 4]

NUMERICAL INTEGRATION (QUADRATURE)

- ADQUAD** Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of functional evaluations required.
- GAUSS** Calculates Gaussian quadrature abscissae and weights relative to a given weight function on a given finite or infinite interval of integration.
- SIMPSN** Integrates a function using Simpson's rule if given at equally spaced abscissae or using interpolatory quadratics if given at unequally spaced abscissae.

CHAPTER 5**SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS**

- AMI** Performs one step of the implicit fourth order Adams-Moulton method. No error control.
- DIFSUB** Performs one variable step using various methods. Good for stiff systems. Good error control
- GEAR** Performs one step using Gear's subroutine, DIFSUB. Built-in error control.
- RKI** Performs one step of the explicit fourth order Runge-Kutta-Gill method. No error control.

CHAPTER 6**SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS**

- BLKTRI** Solves a separable elliptic equation.
- POIS** Solves an elliptic equation with variable coefficients on one derivative.
- PWSCRT** Solves two-dimensional Helmholtz equation in Cartesian coordinates.
- PWSCSP** Solves a two-dimensional Helmholtz equation in spherical coordinates assuming no functional dependence on longitude.
- PWSCYL** Solves two-dimensional Helmholtz equation in cylindrical coordinates.
- PWSSSP** Solves two-dimensional Helmholtz equation on the surface of a sphere.

CHAPTER 7**EVALUATION OF SPECIAL MATHEMATICAL FUNCTIONS**

- BESLIK** Computes modified Bessel functions of the first and second kind for real argument and real order.
- BESLJY** Computes Bessel functions of the first and second kind for real argument and real order.
- CXERFC** Computes the complex complementary error function of a complex argument.
- ELIPE** Computes complete elliptic integrals of the second kind.
- ELIPK** Computes complete elliptic integrals of the first kind.
- EXPINT** Computes exponential integrals.
- HYPER** Computes hyperbolic functions.

CHAPTER 8**FAST FOURIER ANALYSIS**

- FFT** Computes fast Fourier transforms for data vectors of arbitrary length.
- FFTPOW2** Computes fast Fourier transforms for data vectors whose length is a power of two.

CHAPTER 9

STATISTICAL ANALYSIS AND RANDOM NUMBER GENERATION

CNFPRB	Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients.
RGRSN1	Calculates a set of regression coefficients for an unweighted regression model.
RGRSN2	Weights the model with a diagonal matrix of weights.
RGRSN3	Weights the model with the covariance matrix of the random variables.
RGRSN4	Similar to RGRSN3 except that the covariance matrix is banded.
RGRSN5	Weights the model using standard deviations of repeated observations.
RGRSN6	Weights the model using an estimated covariance matrix from repeated observations.
RLHPTS	Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable.
RNDEV	Generates independent random deviates with mean 0 and variance 1.
SPAL	Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data.
SPECFT	Calculates a single power spectrum of an input time series or the cross spectrum of two input time series, including cospectrum, quadrature, coherence, and phase.

[CHAPTER 10]

SPECIAL-PURPOSE INPUT/OUTPUT ROUTINES

BRANRD	Provides buffered random access I/O.
CORFOR	Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted.
READLX	Provides free format data-directed input and program control facilities.
TAPECY	Copies, combines, and edits tapes.
UBLOK	Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries.
UZBLOK	Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries.

[CHAPTER 11]

DATA PROCESSING UTILITY ROUTINES

BSEARCH	Performs binary search of a table of floating point numbers.
CHCONV	Converts characters in one character set to another character set by indexing a conversion table.
CONV360	Converts IBM 360 (EBCDIC) cards to CDC (BCD) cards.
DATE	Computes date information from number of hours after December 31, 1920.
HOURS	Computes hours since December 31, 1920, from date information.

[CHAPTER 11]
(continued)

DATA PROCESSING UTILITY ROUTINES (continued)

- LCKSUM** Provides a fast checksum of data.
- MOVE** Provides a rapid in-core transfer of data.
- UCHAR** Provides a rapid character retrieval facility.

[CHAPTER 12]

COMPUTER GRAPHICS

- AUTOGRAPH** Draws and annotates curves or families of curves.
- CONREC Standard** Contours two-dimensional arrays, labelling the contour lines.
- CONRECQCK** Like CONREC Standard, but faster and smaller because contours are unlabelled.
- CONRECSMTH** Like CONREC Standard, but bigger and slower because contours are smoothed as well as labelled.
- DASHCHAR** Software dashed line package with labelling capability.
- DASHLINE** Like DASHCHAR, but smaller and faster because it has no labelling capacity.
- DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed.
- HAFTON** Halftone (gray scale) pictures from a two-dimensional array.
- ISOSRF** Iso-valued surfaces (with hidden lines removed) from a three-dimensional array.

[CHAPTER 12]
(continued)

COMPUTER GRAPHICS (continued)

ISOSRFHR	Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array.
PWRX	High quality software characters.
PWRY	Simplest software characters.
PWRZ	Three-space characters for use with ISOSRF or SRFACE.
SCROLL	Movie titling package.
SRFACE	Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array.
SUPMAP	Continental outlines and political boundaries in various projections.
VELVEC	Two-dimensional velocity field displayed by drawing arrows from the data locations.

[CHAPTER 13]

FILE MANIPULATION, TEXT EDITING, PROGRAM PREPROCESSING AND DEBUGGING

EDITOR	Program to maintain a tape library of source card files in a PLIB-compatible form.
FIDEL	Pre-compiler for the language PDELAN (an extension of FORTRAN to facilitate implementation of finite difference approximations to partial differential equations).
FLEX	File management of COSYed PLIB card files.
FRED	Pre-compiler for FORTRAN programs providing conditional compilation, macros, debugging aids, evaluation of subscript expressions and renumbering of statement labels.

**ALPHABETICAL LISTING
OF SUBROUTINES****A****ADQUAD**

Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of functional evaluations required. (Volume I, Chapter 4)

AMI

Performs one step of the implicit fourth order Adams-Moulton method. No error control. (Volume II, Chapter 5)

AUTOGRAPH

Draws and annotates curves or families of curves. (Volume III, Chapter 12)

B**BDSL**

Solves a banded linear system. (Volume I, Chapter 3)

BESLIK

Computes modified Bessel functions of the first and second kind for real argument and real order. (Volume II, Chapter 7)

BESLJY

Computes Bessel functions of the first and second kind for real argument and real order. (Volume II, Chapter 7)

BLKTRI

Solves a separable elliptic equation. (Volume II, Chapter 6)

BND3

Solves a tridiagonal linear system using Gaussian elimination with partial pivoting. (Volume I, Chapter 3)

BRANRD

Provides buffered random access I/O. (Volume III, Chapter 10)

BSEARCH

Performs binary search of a table of floating point numbers. (Volume III, Chapter 11)

BSL1NT

Performs one-dimensional Bessel interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

BSL2NT

Performs two-dimensional Bessel interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

C

- CHCONV** Converts characters in one character set to another character set by indexing a conversion table. (Volume III, Chapter 11)
- CHLSLV** Solves a real, symmetric, positive definite linear system by Cholesky decomposition. (Volume I, Chapter 3)
- CNFPRB** Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients. (Volume II, Chapter 9)
- CONV360** Converts IBM 360 (EBCDIC) cards to CDC (BCD) cards. (Volume III, Chapter 11)
- CONREC Standard** Contours two-dimensional arrays, labelling the contour lines. (Volume III, Chapter 12)
- CONRECQCK** Like CONREC Standard, but faster and smaller because contours are unlabelled. (Volume III, Chapter 12)
- CONRECSMTH** Like CONREC Standard, but bigger and slower because contours are smoothed as well as labelled. (Volume III, Chapter 12)
- CORFOR** Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted. (Volume III, Chapter 10)
- CUBSPL** Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives. (Volume I, Chapter 2)
- CURV** Performs one-dimensional interpolation, differentiation, and integration using splines under tension. (Volume I, Chapter 2)
- CXERFC** Computes the complex complementary error function of a complex argument. (Volume II, Chapter 7)

D

- DASHCHAR** Software dashed line package with labelling capability. (Volume III, Chapter 12)
- DASHLINE** Like DASHCHAR, but smaller and faster because it has no labelling capacity. (Volume III, Chapter 12)
- DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed. (Volume III, Chapter 12)
- DATE** Computes date information from number of hours after December 31, 1920. (Volume III, Chapter 11)

D *(continued)*

- DBNDZRO** Computes a posteriori error bounds and improvements to roots of polynomials. (Volume I, Chapter 1)
- DCPOLY** Finds all the roots of a polynomial with complex coefficients using double precision. (Volume I, Chapter 1)
- DIFSUB** Performs one variable step using various methods. Good for stiff systems. Good error control. (Volume II, Chapter 5)
- DRPOLY** Finds all the roots of a polynomial with real coefficients using double precision. (Volume I, Chapter 1)

E

- EDITOR** Program to maintain a tape library of source card files in a PLIB-compatible form. (Volume III, Chapter 13)
- EIGCFA** Computes all the eigenvalues and selected eigenvectors of a general complex matrix. (Volume I, Chapter 3)
- EIGHFS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix. (Volume I, Chapter 3)
- EIGRFA** Computes all the eigenvalues and selected eigenvectors of a general real matrix. (Volume I, Chapter 3)
- EIGSFM** Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric matrix. (Volume I, Chapter 3)
- EIGSFS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix. (Volume I, Chapter 3)
- EIGSTM** Computes the algebraically smallest or largest M eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix. (Volume I, Chapter 3)
- EIGSTS** Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix. (Volume I, Chapter 3)
- ELIPE** Computes complete elliptic integrals of the second kind. (Volume II, Chapter 7)
- ELIPK** Computes complete elliptic integrals of the first kind. (Volume II, Chapter 7)

E *(continued)*

EXPINT Computes exponential integrals. (Volume II, Chapter 7)

F

FFT Computes fast Fourier transforms for data vectors of arbitrary length. (Volume II, Chapter 8)

FFTPOW2 Computes fast Fourier transforms for data vectors whose length is a power of two. (Volume II, Chapter 8)

FIDEL Pre-compiler for the language PDELAN (an extension of FORTRAN to facilitate implementation of finite difference approximations to partial differential equations). (Volume III, Chapter 13)

FLEX File management of COSYed PLIB card files. (Volume III, Chapter 13)

FRED Pre-compiler for FORTRAN programs providing conditional compilation, macros, debugging aids, evaluation of subscript expressions and renumbering of statement labels. (Volume III, Chapter 13)

G

GAUSS Calculates Gaussian quadrature abscissae and weights relative to a given weight function on a given finite or infinite interval of integration. (Volume I, Chapter 4)

GEAR Performs one step using Gear's subroutine, DIFSUB. Built-in error control. (Volume II, Chapter 5)

H

HAFTON Halftone (gray scale) pictures from a two-dimensional array. (Volume III, Chapter 12)

HOURS Computes hours since December 31, 1920, from date information. (Volume III, Chapter 11)

HRM1NT Performs one-dimensional Hermite interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

HRM2NT Performs two-dimensional Hermite interpolation of selected order for values and derivatives. (Volume I, Chapter 2)

HSHSLV Solves a real overdetermined linear system using Householder transformations. (Volume I, Chapter 3)

HYPER Computes hyperbolic functions. (Volume II, Chapter 7)

I

- INVMTX** Computes the inverse of a general real matrix using Gaussian elimination with full pivoting. (Volume I, Chapter 3)
- ISOSRF** Iso-valued surfaces (with hidden lines removed) from a three-dimensional array. (Volume III, Chapter 12)
- ISOSRFHR** Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array. (Volume III, Chapter 12)

K

- KURV** Performs interpolation of a parameterized curve in the plane using splines under tension. (Volume I, Chapter 2)
- KURVP** Performs interpolation of a parameterized closed curve in the plane using splines under tension. (Volume I, Chapter 2)

L

- LCKSUM** Provides a fast checksum of data. (Volume III, Chapter 11)
- LINEQSV** Solves a real linear system using Gaussian elimination with partial pivoting. (Volume I, Chapter 3)

M

- MOVE** Provides a rapid in-core transfer of data. (Volume III, Chapter 11)

P

- POIS** Solves an elliptic equation with variable coefficients on one derivative. (Volume II, Chapter 6)
- PWRX** High quality software characters. (Volume III, Chapter 12)
- PWRY** Simplest software characters. (Volume III, Chapter 12)
- PWRZ** Three-space characters for use with ISOSRF or SRFACE. (Volume III, Chapter 12)
- PWSCRT** Solves two-dimensional Helmholtz equation in Cartesian coordinates. (Volume II, Chapter 6)
- PWSCSP** Solves a two-dimensional Helmholtz equation in spherical coordinates assuming no functional dependence on longitude. (Volume II, Chapter 6)

P *(continued)*

- PWSCYL** Solves two-dimensional Helmholtz equation in cylindrical coordinates. (Volume II, Chapter 6)
- PWSSSP** Solves two-dimensional Helmholtz equation on the surface of a sphere. (Volume II, Chapter 6)

Q

- QURV** Performs interpolation of a parameterized curve in space using splines under tension. (Volume I, Chapter 2)

R

- READLX** Provides free format data-directed input and program control facilities. (Volume III, Chapter 10)
- RGRSN1** Calculates a set of regression coefficients for an unweighted regression model. (Volume II, Chapter 9)
- RGRSN2** Weights the model with a diagonal matrix of weights. (Volume II, Chapter 9)
- RGRSN3** Weights the model with the covariance matrix of the random variables. (Volume II, Chapter 9)
- RGRSN4** Similar to RGRSN3 except that the covariance matrix is banded. (Volume II, Chapter 9)
- RGRSN5** Weights the model using standard deviations of repeated observations. (Volume II, Chapter 9)
- RGRSN6** Weights the model using an estimated covariance matrix from repeated observations. (Volume II, Chapter 9)
- RKI** Performs one step of the explicit fourth order Runge-Kutta-Gill method. No error control. (Volume II, Chapter 5)
- RLHPTS** Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable. (Volume III, Chapter 9)
- RNDEV** Generates independent random deviates with mean 0 and variance 1. (Volume III, Chapter 9)
- RTNI** Finds a root of a given non-linear equation. (Volume I, Chapter 1)

S

- SCROLL** Movie titling package. (Volume III, Chapter 12)
- SIMPSN** Integrates a function using Simpson's rule if given at equally spaced abscissae or using interpolatory quadratics if given at unequally spaced abscissae. (Volume I, Chapter 4)
- SPAL** Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data. (Volume II, Chapter 9)
- SPECFT** Calculates a single power spectrum of an input time series or the cross spectrum of two input time series, including cospectrum, quadrature, coherence, and phase. (Volume II, Chapter 9)
- SPLPAK** Performs least squares fitting of multidimensional cubic splines to arbitrarily located data. (Volume I, Chapter 2)
- SRFACE** Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array. (Volume III, Chapter 12)
- SUPMAP** Continental outlines and political boundaries in various projections. (Volume III, Chapter 12)
- SUPRLS** Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core. (Volume I, Chapter 3)
- SURF** Performs two-dimensional interpolation using a bi-spline under tension. (Volume I, Chapter 2)
- SVDSLV** Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core. (Volume I, Chapter 3)

T

- TAPECY** Copies, combines and edits tapes. (Volume III, Chapter 10)
- TRDI** Solves a diagonally dominant tridiagonal linear system using Gaussian elimination without pivoting. (Volume I, Chapter 3)
- TRDIP** Solves a diagonally dominant periodic tridiagonal linear system using Gaussian elimination without pivoting. (Volume I, Chapter 3)

T *(continued)***TRIANGLE**

Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane. (Volume I, Chapter 2)

U**UBLOK**

Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries. (Volume III, Chapter 10)

UCHAR

Provides a rapid character retrieval facility. (Volume III, Chapter 11)

UZBLOK

Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries. (Volume III, Chapter 10)

V**VELVEC**

Two-dimensional velocity field displayed by drawing arrows from the data locations. (Volume III, Chapter 12)

10

INPUT/OUTPUT

BRNRD

CORFOR

READLX

TAPECY

UBLOK

UZBLOK

SUBROUTINE BRNRD (INDEX,FWA,N)**LATEST REVISION** September 30, 1973**PURPOSE**

BRNRD provides buffered I/O operations to the 'RANDOM FILE'. The 'RANDOM FILE' is a set of files that may be accessed by the user in any order. Each file contains one record and is labeled by an index that is specified by the user. The routine has a true buffering capability by using two I/O channels. Channel selection is based upon the value of the index parameter being odd or even. Five entry points are provided in this routine. BRNRD(BRANWT) initiates a read (write) of a file. BRANCK determines that a read(write) has been successfully completed. BRANST checks the status of a read(write) and tells the user whether the I/O is complete or not. It does not wait for a completion before returning to the calling program. BRANRL is used on the 7600 to release LCM space.

ACCESS CARDS

*ASCENT,S=ULIB,N=BRNRD
*COSY

NOTE

BRANRD is also on system binary library. That binary form of BRANRD should be used unless the user wishes to modify the code.

USAGE

CALL BRANRD (INDEX,FWA,N)
CALL BRANWT (INDEX,FWA,N)
CALL BRANCK (INDEX)
CALL BRANST (INDEX,NSTAT)
CALL BRANRL

ARGUMENTS

On Input

INDEX

The record index key. Any 60 bit combination except +0 or -0.

FWA

Address in central memory of first word of record.

N

Number of 60 bit central memory words to be transferred.

On Output

INDEX, FWA, N

Unchanged. See "SPECIAL CONDITIONS".

NSTAT

= -1 I/O is not complete.

= 0 I/O has been completed successfully.

ENTRY POINTS

BRANRD, BRANWT, BRANCK, BRANST, BRANRL

SPECIAL CONDITIONS

After each call to BRANWT or BRANRD for an even (odd) record index key, a call must be made to BRANCK using the same even (odd) record index key to check for successful I/O completion before another operation is started with another even (odd) record index key. On the 7600, a record index key table with space for 200 unique keys is put into Large Core Memory (LCM). Initial records are also put into LCM, to the extent of available LCM, and subsequent records are sent to the 7600 disk files. If the subroutine encounters an error condition, it prints a message and terminates program execution. The messages are printed by a system routine Q8QERR.

The messages are:

ATTEMPT TO READ UNWRITTEN RECORD
 ATTEMPT TO USE RECORD WITH A LENGTH .GT. 1ST USE LENGTH
 ATTEMPTED ANOTHER OPERATION WITHOUT CHECKING LAST
 CANNOT RESET FILE
 CANNOT SETUP FILE
 EOF ENCOUNTERED
 NAME MUST BE NONZERO
 RECORD LENGTH MUST BE .GT. 0
 UNSUCCESSFUL READ OR WRITE

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Gilbert Green, NCAR, Boulder, Colorado 80303

LANGUAGE

ASCENT

HISTORY

Documented October 1, 1973

SPACE REQUIRED

254₈ = 172₁₀ locations

TIMING

6600: Each drum access required ~17 milliseconds of PP time for each buffered read or write

7600: Each disk access required ~85 milliseconds of PP time for those records sent to the 7600 disk files instead of being contained in LCM.

In either case, the actual transfer time for the record is negligible except for very large records.

PORTABILITY

Not portable.

REQUIRED RESIDENT ROUTINES

Q8QERR

PROCEDURE

BRANRD provides a means of temporary buffered random access storage on two separate I/O channels for a program. Records with even valued record index keys are sent to the first channel; those with odd valued record index keys are sent to the second channel. Since the two channels can transmit simultaneously, it is possible to be reading on one channel while writing on the other channel. Record index keys can be any value except ± 0 . In general, left justified, blank filled Hollerith names will be odd because the low order bit of the internal representation of a blank, 55₈ = 101101₂, is a 1 and integers in floating point representation will be even because bit 0 will be a zero due to normalization of floating point numbers.

Before a new operation is started with the same or a different even (odd) record index key, the success and completion of the preceding even (odd) operation should be determined by a call to BRANCK.

Records may be any length greater than zero. Subsequent reads and rewrites of an individual record may be of different lengths with the stipulation that the length used in initial write of any record will be the maximum length used in reading or writing that particular record.

If the program is executing on the 7600, an initial table with space for 200₁₀ record index key entries is established. As each initial write of a record is initiated, additional LCM is requested and the records are kept in LCM until no more user LCM space is available and then subsequent records go to the 7600 disks using the even/odd record index key channel assignment algorithm. On the 6600 records are sent directly to the drums and no table is created.

If more than 200 record index keys are needed, the value of the EQUated symbol, NRECS, can be changed in the ULIB version of BRANRD to the needed number. The program librarian on duty will be glad to assist in this procedure if help is needed.

If the program is executing on the 7600, a call to BRANRL will give all of the programmer requested LCM back to the system. This is useful in reducing CRU charges or for processing a second data set using the same record index keys, but different (longer) record lengths. On the 6600, a call to BRANRL acts as a NOP (no operation).

SUBROUTINE CORFOR (LIM,LTH,LREC,NOPT,NFILE,IBUF)

DIMENSION OF ARGUMENTS IBUF(LIM)

LATEST REVISION September 1969

PURPOSE CORFOR allows the user to copy a FORTRAN-written binary tape with read errors. Records with bad parity may be copied or deleted. The tape to be copied must have been written on the NCAR system.

ACCESS CARDS *FORTRAN,S=ULIB,N=CORFOR
 *COSY

 *FORTRAN,S=ULIB,N=RDFOR
 *COSY

 *ASCENT,S=ULIB,N=LCKSUM
 *COSY

USAGE CALL CORFOR (LIM,LTH,LREC,NOPT,NFILE,IBUF)

ARGUMENTS

On Input

LIM

Dimension of IBUF.

LTH

Record length.

= 0 for normal copy

= N to force all records to length N.

LREC

Number of logical records to copy.

NOPT

Bad record flag

= 0 to copy all records

= 1 to delete bad records

NFILE

Number of files to copy

On Output

IBUF

Buffer to contain records during the copy.

ENTRY POINTS

CORFOR, RDFOR, MOVE

SPECIAL CONDITIONS

Maximum physical binary record length is 512 words. Input tape is assumed to have been written on the NCAR system.

FORTTRAN binary write segments the logical record into 512 word physical records.

COMMON BLOCKS

None.

I/O

FORTRAN logical unit 1--tape to be copied.

FORTRAN logical unit 8--output tape.

All I/O messages are printed from subroutines CORFOR and RDFOR:

CORFOR

For a record with a parity error, CORFOR prints

*ERROR AT LOGICAL RECORD NUMBER nn LENGTH nn
and then prints the first two words and the last word read
into the buffer.

For the first 500 records, CORFOR prints the word *REC*
then the input record number, the output record number,
number of words in the record, the status on the record
read and the two initial words, plus the final word read
into the buffer.

For a multifile reel, CORFOR prints the following after
an EOF has been written on the output tape:

*nn RECORDS IN, nn RECORDS OUT. LENGTHS of nn TO
mm WORDS. FILE NO nn*

After NFILE files have been copied, CORFOR prints

JOB COMPLETE

I/O*(continued)***RDFOR**

For records one or two words long, RDFOR prints
 SHORT PHYSICAL RECORD - REC nn, LENGTH nn
 (FORTRAN binary records must be at least 3 words in length.)

If number of words written does not match number of words
 read, RDFOR prints

*LENGTH ERROR PHYS REC NO nn, LENGTH WRITTEN nn,
 LENGTH READ nn*

If the sequence number in the binary record label does not
 match the count of records read, RDFOR prints

PHYSICAL SEQ ERROR COUNT = nn, SEQ = nn .

If an EOF was sensed before completing a logical record
 read on the last read, RDFOR prints

EOF IN MIDDLE OF REC nn .

If a parity error was sensed on the last read, RDFOR
 prints

PARITY ERROR, REC = nn .

If the CHECKSUM computed after the record was read does not
 match the CHECKSUM computed when the record was written,
 RDFOR prints

CSUM ERROR IN REC NO nn,
 CSUM WRITTEN = nn,
 CSUM READ = nn .

PRECISION

Single

REQUIRED ULIB ROUTINES	LCKSUM
SPECIALIST	Marie Working, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN, ASCENT
HISTORY	Written by Dennis Joseph, September 1969. Standardized in February 1974.
ALGORITHM	None
SPACE REQUIRED	$7332_8 = 3802_{10}$
ACCURACY	Not applicable
PORTABILITY	Not portable
REQUIRED RESIDENT ROUTINES	RDTAPE, IOWAIT, GBYTE

LEXICAL READ PACKAGE

LATEST REVISION November 1973

PURPOSE To provide simplified, general-purpose data-directed I/O and program control language facilities for FORTRAN programs in modelling and testing applications where it is difficult to design a sufficiently flexible and comprehensive communications interface in advance.

USAGE Initially, the user constructs a namelist dictionary, associating Hollerith identifiers with selected program elements which he wishes to make accessible via the "READLX" interpreter. Both internal variable locations and external subroutine entry points are acceptable, as follows:

```
EXTERNAL SUBRTN
DIMENSION VARIABLE(7)
CALL LEXCON (VARIABLE ,7HVARIBLE  NVALUS)
CALL LEXCON (SUBRTN   ,7HSUBRTIN  NPARMS)
```

USAGE*(continued)*

Subsequently, simple assignment statements and subroutine calls of the forms

```
VARIABLE = LIST,OF,NEW,VALUES
SUBRTN (LIST,OF,ARGUMENTS)
```

may be interpreted directly as input, without prior specification of number, order, or format, via the single call

```
CALL READLX (LUNIT,KEND)
```

During interpretation, new values are assigned to variables and subroutines are called entirely asynchronously to ordinary flow of control in the user program. The "READLX" interpreter itself retains program control until a suitable "END" signal is encountered in the input text.

ACCESS CARDS

```
*FORTRAN,S=ULIB,N=LEXICAL
*COSY
```

ARGUMENTS

On Input
for LEXCON

```
Call LEXCON (IVAR,IDENT,INDX)
```

IVAR

= Internal variable or external subroutine in the user program to be made accessible via "READLX".

IDENT

= Hollerith identifier to be associated with IVAR (several IDENTIS may be given as synonyms for one IVAR, and IDENTIS may be redefined as often as necessary, but they must all be unique).

On Output
for LEXCON

INDX

= Counter variable in user program to receive number of new values assigned to a variable, or number of parameters passed to a subroutine during "READLX" interpretation.

On Input
for READLX

Call READLX (LUNIT,KEND)

LUNIT

= Logical unit number of "READLX" input.

On Output
for READLX

KEND

= End-of-read/error flag

If -ve Error encountered during read

If 0, Successful read, terminated by "END"

If +1, Successful read, terminated by "ENDOFREAD"

If +2, Successful read, terminated by "ENDOFDATA"

If +3, Successful read, terminated by "ENDOFCASE"

READLX CARD INPUT

Card Format

Entirely free-format between columns 2 through 72 inclusive. Continuation is implied by either 1) a trailing comma on the preceding card, or 2) a leading comma on the continued card. Several statements, separated by dollar signs (\$), may also be coded on a single card. The letter "C" punched in column 1 will cause the entire line to be treated as a comment.

Variable Names

Up to seven alphameric characters, beginning with a letter, and optionally qualified by a single integer (or octal) constant subscript. No mode or dimension is implied by variable names, and it is the user's responsibility to assure correct data types and to avoid oversteering arrays.

Constant Data

Both numeric and character data types, as follows:

- | | | |
|----------------|-----------------------------------|---------------|
| 1. Integer | 0, 1, -2, +3 | |
| 2. Real | 0., .1, -2.718, +3.14159 | |
| 3. Exponential | 0E0, 1.E-1, -.2718E+1, +314159E-5 | |
| 4. Octal | 0B, 77B, 77777777777777777700B | |
| 5. Hollerith | 1H0, 9HHOLLERITH | (70 char max) |
| 6. String | '0', 'QUOTED STRING' | (70 char max) |

Constant values of any and/or mixed types may be assigned to any variable: no mode conversion or bounds checking is performed.

Variable Assignment

These statements have the form

```
VAR = VALIST
```

where VALIST is a list of constant and/or variable data items separated by commas (if more than 1). Elision of items may be indicated by succeeding commas without intervening symbol, and replication of a single item by a repetition count prefixed to the value by a colon (:, or equivalently ..).

Subroutine Call

These statements have the form

```
SUBRIN (ARGLIST)
```

where ARGLIST is a list of (up to 7) constant and/or variable data items separated by commas. Constant arguments are passed by value to the subroutine, while symbolic variables (in the user program) are passed by reference, as in FORTRAN.

Read Termination

Interpretation continues until an identifier beginning with the 3 characters "END" is encountered, or an error occurs. On return from the "READLX" interpreter, the coded value of this terminator, or error code, is placed in a parameter for inspection.

EXAMPLES:

In User Program

```

EXTERNAL SUM
EXTERNAL LINSOLV
EXTERNAL WRITLX

DIMENSION VECTOR (3),MATRIX(3,3),RESULT(3)

CALL LEXCON (SUM      ,7HSUM      ,DMY)
CALL LEXCON (LINSOLV ,7HLINSOLV ,DMY)
CALL LEXCON (WRITLX  ,7HWRITLX  ,DMY)

CALL LEXCON (SCALAR  ,6HSCALAR,DMY )
CALL LEXCON (VECTOR(1),6HVECTOR,LENGTH)
CALL LEXCON (VECTOR  ,6HVECT  ,LENGTH)
CALL LEXCON (MATRIX  ,6HMATRIX,DUMMY )

```

On Input Stream

```

SCALAR      = 0
VECTOR      = 1.,2.,3.
LETTER(1)   = 1HA$LETTER(2)=1HB$---etc.
DIGITS      = '0', '1'---etc.
COMMENT 3*3 Identity Matrix
MATRIX      = 1,0,0
              0,1,0
              0,0,1
MATRIX      = 9:0$MATRIX=1,,,,1,,,,1
INDEF       = 6000000000000004000000B
ARRAY       = 100..INDEF

END

LINSOLV (MATRIX,3,3,VECTOR,RESULT)
SUM (RESULT,3,SCALAR)
WRITLX ('(F10,2)',RESULT)

END-OF-EXAMPLE

```

ENTRY POINTS	LEXCON,READLX LEXRTN,LXCARD,LXCHAR,LEXFMT,LEXERR
COMMON BLOCKS	/LXDICT/300 octal (contains Lexical Dictionary)/LXINTC/300 octal (contains Internal Parameters)
I/O	Formatted input from LUNIT, echoed and error message output on system printer.
PRECISION	Not applicable
REQUIRED ULIB ROUTINES	None
SPECIALIST	Jordan Towner Hastings, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Initially written by R. V. Helgason. Extended and standardized November 1973.
SPACE REQUIRED	3000 ₈ including COMMON BLOCKS, plus FORTRAN formatted I/O routines.

ACCURACY

Constant values interpreted by READLX have internal machine representations identical to those produced by formatted Reads.

TIMING

About 3 times slower than formatted Reads for the same amount of data.

PORTABILITY

This routine is immediately portable to CDC 6000/7000 sites. Other users must provide a LOC function in LEXCON and READLX, verify subroutine/parameter passing conventions in LEXRTN, and recode for ENCODE/DECODE statements in LEXFMT routine.

**REQUIRED RESIDENT
ROUTINES**

INPUTC, INPUTS, KRAKER, OUTPTC, OUTPTS, KODER

PROGRAM TAPECY

LATEST REVISION Febraury 1974

PURPOSE TAPECY is a driver (main) program that copies, combines and edits tapes.

ACCESS CARDS *FORTRAN,S=ULIB,N=TAPECY
 *COSY

USAGE Data control cards must be supplied to be read by subroutine COPY. Tape drives must be assigned.

ENTRY POINTS COPY, USER, USERF

SPECIAL CONDITIONS • Logic may be added by users to subroutine USER to edit/select records by examining data in the record. Without additional logic, the record will be copied. USER is called after a record is read, except for a record that is read with a SKIP control card or a record that has a parity error and the error option on the control card has been set to 1.

SPECIAL CONDITIONS

(continued)

- USERF will copy ends of files as read on input tape unless logic is added to delete end of files. USERF is not called unless the control word on the control card is CFILE and an end of file occurs.
- If records on the tape to be copied are longer than 1000 60-bit words, then LTH must be set to the longest record length and BUF must be dimensioned by 2*LTH in main program TAPECY.
- Any record read with a word length of zero will be ignored (not counted as a record read and not processed further). A message will, however, be printed. A record indicated as zero length is usually noise in a record gap.
- If an end of tape is encountered on a read and the indicated record length is zero, the read will be treated as an end of file. If the record length is greater than zero, it will be treated as a record.
- If an end of tape is encountered on a write, that record will not be copied.

COMMON BLOCKS

None

I/O

Tape Assignment

Input and output tape drives must be assigned to agree with data control cards.

Data Control Cards (Read by Copy)

Program Control is accomplished by using control cards in the data section of the program. Each control card is processed sequentially and is printed before it is processed.

Format

<u>Card</u>	<u>Field</u>
<u>Column</u>	
1-5	Control word (left justified)
6-10	NREC = number of records or files
21-22	UNIT1 = A logical tape unit (right justified)
27-28	UNIT2 = A logical tape unit (right justified)
38	Mode of tape on UNIT1
40	Mode of tape on UNIT2
48	Type of tape on UNIT1
50	Type of tape to be written on UNIT2
57	Error option
	1 = Copy only records without parity errors
	2 = Copy all records

Control Words (column 1-5)

REM

Remark card which contains any remarks in columns 6-80.

REW

Rewind the unit specified in the field UNIT1
(column 21-22).

WEOF

End file on the unit specified in the field UNIT1
(column 21-22).

COPY

Copy NREC records (or one file, whichever occurs first)
from UNIT1 to UNIT2 with the modes, types, and error
option as specified.

I/O

(continued)

CFILE

Copy NREC files as for COPY. The copy will continue until NREC files have been copied or terminated in the USERF routine (NF = 2 or 3).

SKIP

Skip NREC records (or one file, whichever comes first) on UNIT1. The output parameters must be specified even though they are not used.

UNLOAD

Rewind and unload UNIT1.

DONE

Terminates job.

MODE

- 0 = Even parity (BCD mode), no character conversion.
- 1 = Odd parity (binary mode), no character conversion.
- 2 = Even parity (BCD mode), conversion of external BCD to display code (or vice versa if writing a record).

NTYPE

- 0 or 1 = Read (or write) a record in the NCAR system format.
- 2 = Read a record from another computer.

Note: Add 4 to the above options for NTYPE to ignore a read parity error. Thus type 6 is the same as type 2, but with no reread on parity errors.

SUBROUTINE ARGUMENTS

SUBROUTINE COPY (BUF,LTH) These arguments are set by TAPECY.

On Input

BUF

The buffer array for tape records being copied. The dimension of this array is equal to LTH*2 to provide for a double buffered copy.

LTH

Equal to or longer than the number of words in the longest record. (Set to 1000.)

SUBROUTINE USER (IBUF,IWORDS,NFILE,NREC,ISTATE,NN) All arguments are set by subroutine COPY except NN. Called by COPY after reading each record unless

- End of file is read.
- Error option on control card is set to 1 and current record has a parity error.
- Control word is SKIP.

On Input

IBUF

Address of the first word of the current record.

IWORDS

Length of the current record.

NFILE

Number of the current file read under control of the current control card.

NREC

Number of records read in the current file.

ISTATE

State of the current read

- = 0 Good read.
- = 1 End of file.
- = 2 Parity error.
- = 3 End of tape.

On Output

NN

A flag set in USER to determine if the record should be copied. It is set to 1 by the unmodified version of USER on ULIB. USER must be modified if logic is desired to determine whether a record should be copied.

= 0 Do not copy.

= 1 Copy.

SUBROUTINE USERF (NRSUM,NRECS,NFILE,NF) All arguments are set by COPY except NF.

Called by COPY only when control word is CFILE and an end of file occurs.

On Input

NRSUM

The total number of records read under control of current control card.

NRECS

The number of records in the file just completed.

NFILE

The number of the file just completed.

On Output

NF

The flag set for action taken in this subroutine. The version on ULIB simply sets NF = 1, except when 2 consecutive end of files are encountered, NF is set to 3. USERF should be modified if logic is desired to determine whether an end of file should be written and/or the next control card should be read. NF options are:

= 0 Continued copy with next file and do not write an end of file.

= 1 Same as 0, but write an end of file.

= 2 Go to next control and do not write an end of file.

= 3 Go to next control card after writing an end of file.

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Dean Frey, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Written by Roy Jenne; standardized February 1974.

Method

TAPECY is used to copy, merge and edit tapes. Copying and merging is controlled by the use of data control cards. The option of copying or deleting records with parity errors may also be determined by the use of control cards. Other editing is accomplished by modifying the subroutines USER and USERF. If USER and USERF are not modified, a simple copy of the tape will be made.

USER makes available the data from the current record so that information in the record such as time/data or any other parameter may be scanned and the record copied or deleted by setting the argument NN to 1 or 0.

USERF may be modified to delete end of files on a selective basis (called only when control word is CFILE).

EXAMPLES

Example 1

Suppose we wish to simply copy a binary tape (written at NCAR) with 5 files and the maximum record length is less than 1000 60-bit words, then the deck would be

```

*JOB
*ASSIGN,C1410=1,R
*ASSIGN,C2500=8
*LIMIT,T=,PT=
*FORTRAN,S=ULIB,N=TAPECY
*COBY
*RUN
REN                1
REN                8
CFILE 5            1  8      1 1    0 0    2
MEOF
MEOF                8
DONE
*END
    
```

If the tape were BCD mode, the only change would be card

```

CFILE 5            1  8      1 1    0 0    2
                TO
CFILE 5            1  8      0 0    0 0    2
    
```

If only good records (no parity error) were to be copied, the only change would be card

```

CFILE 5            1  8      1 1    0 0    2
                TO
CFILE 5            1  8      1 1    0 0    1
    
```

If the tape was not written on the NCAR system, the only change would be card

```

CFILE 5      1  8      11    00    2
      TO
CFILE 5      1  8      11    20    2

```

Example 2

The computer example which follows is output from a tape copy which, for reason of illustration, is unusually complicated and contains a high number of errors.

This procedure will copy two files from the tape assigned on logical unit 1, to the tape assigned on logical unit 8. Then it would copy 50 records, skip 25 records, copy to the next end of file and then copy the next file. The print lines beginning with control words are images of the control cards while the other print lines indicate the results of the previous control cards. Note that comments may be used on the control cards to clarify its operation.

The first file had TROUBLE ON A WRITE on record 57. The TROUBLE ON A WRITE message indicates number of record in, number of record out, length in, length out, and status of the write. (A 0 is a good write, 2 is trouble on writing (probably due to a bad spot on the tape, and a 3 indicates an end of tape.) The second file had a parity error on record number 16. The first file contained 138 records and the second 97. Since the copy 50 record had the error option set to 1 and encountered a parity error at record number 31, that record was not copied. The first copy 9999 record card was terminated when an end of file was encountered

EXAMPLES
(continued)

Example 2 (continued)

after 37 records, each 250 words long. The second copy 9999 record card was terminated when an end of tape was encountered on the input tape after 40 records. Since the record length was zero, the end of tape was treated as an end of file.

At the end, two end of files were written on the output tape.

REH COPY OUTPUT EXAMPLE

REH CARD COLUMNS

REH 10 22 28 40 50 57

REH 01

REH 08

CFILE 2 FLS FROM 1 TO 8 MODE 1,1 TYPE 0,0 ERR 2

<p> TROUBLE ON A WRITE 57 57 400 400 2 158 RECORDS READ WITH LENGTHS OF 400 TO 400 WORDS. EOF WRITTEN ON UNIT 8 PARITY ERROR IN RECORD 16 97 RECORDS READ WITH LENGTHS OF 300 TO 400 WORDS. EOF WRITTEN ON UNIT 8 </p>	<p> 158 RECORDS COPIED FILE NO 1 TOTAL READ 158 97 RECORDS COPIED FILE NO 2 TOTAL READ 255 </p>
<p> COPY 50 REC FROM 1 TO 8 MODE 1,1 TYPE 0,0 ERR 1 PARITY ERROR IN RECORD 31 50 RECORDS READ WITH LENGTHS OF 400 TO 400 WORDS. </p>	<p>49 RECORDS COPIED FILE NO 0 TOTAL READ 50</p>
<p> SKIP 25 REC FROM 1 TO 8 MODE 1,1 TYPE 0,0 ERR 2 25 RECORDS READ WITH LENGTHS OF 250 TO 250 WORDS. </p>	<p>0 RECORDS COPIED FILE NO 0 TOTAL READ 25</p>
<p> COPY 9999 REC FROM 1 TO 8 MODE 1,1 TYPE 0,0 ERR 2 37 RECORDS READ WITH LENGTHS OF 250 TO 250 WORDS. </p>	<p>37 RECORDS COPIED FILE NO 0 TOTAL READ 37</p>
<p>HEOF 8</p>	
<p> COPY 9999 REC FROM 1 TO 8 MODE 1,1 TYPE 0,0 ERR 2 EOT FOUND AFTER RECORD 40 ,INDICATED RECORD LENGTH= 0 40 RECORDS READ WITH LENGTHS OF 250 TO 250 WORDS. </p>	<p>40 RECORDS COPIED FILE NO 0 TOTAL READ 40</p>
<p>HEOF 8</p>	
<p>HEOF 8</p>	
<p>DONE</p>	

Example 3

This procedure will copy five files from the tape assigned on logical unit 1 to the tape on logical unit 8. The first file had an indicated zero record length on a record after number 94. This "indicated record" was not counted as a record read and was not copied. The CFILE control card was terminated during the fourth file when an end of tape was encountered on the output tape after 500 records. The job was terminated at this point and the next control card was not read. Normally, an estimate of the tape capacity should be used to avoid trying to write over an end of tape mark.

```

REN          01
REN          08
CFILE  5 FLS FROM 1 TO 8  MODE 1,1 TYPE 0,0 ERR 2
AN INDICATED ZERO LENGTH RECORD AFTER RECORD 94 NOT COUNTED AS RECORD
  638 RECORDS READ WITH LENGTHS OF 460 TO 460 WORDS.      638 RECORDS COPIED  FILE NO  1  TOTAL READ  638
  EOF WRITTEN ON UNIT 8
  742 RECORDS READ WITH LENGTHS OF 460 TO 460 WORDS.      742 RECORDS COPIED  FILE NO  2  TOTAL REAC 1380
  EOF WRITTEN ON UNIT 8
  983 RECORDS READ WITH LENGTHS OF 460 TO 460 WORDS.      983 RECORDS COPIED  FILE NO  3  TOTAL READ 2363
  EOF WRITTEN ON UNIT 8
TROUBLE ON A WRITE 501 500 460 460 3
  501 RECORDS READ WITH LENGTHS OF 460 TO 460 WORDS.      500 RECORDS COPIED  FILE NO  4  TOTAL READ 2864
FIRST CARD NOT PROCESSED  WEOF          8

```


SUBROUTINE UBLOK (ITAPE,MODE,KTYPE,IBUF,KMAX,KBLOK,ICD,IEOF)**DIMENSION OF
ARGUMENTS**

ICD(KBLOK),IBUF(KMAX)

LATEST REVISION

February 1967

PURPOSE

UBLOK unblocks *fixed length* BCD or binary logical records (such as Card Images) from *larger* physical records on a "blocked" tape. Each logical record must be an integer multiple of 60 bits long. One or more logical records must be contained in a physical record and logical records must not overlap into the next physical record. See UZBLOK for unblocking records which are a fixed number of bits in length.

ACCESS CARDS*FORTRAN,S=ULIB,N=UBLOK
*COSY**USAGE**

CALL UBLOK (ITAPE,MODE,KTYPE,IBUF,KMAX,KBLOK,ICD,IEOF)

ARGUMENTS

On Input

ITAPE

Input tape logical unit number.

KBLOK

Number of 60-bit words in each logical record.

KMAX

Maximum physical record size plus one word.

MODE

Mode for tape record

= 0 BCD, no character conversion

= 1 BIN; no character conversion

= 2 BCD, conversion from external BCD to display code.

= 3 BIN, conversion from external BCD to DPC.

KTYPE

Type for tape read

IBUF

On Output

ICD

Output buffer for unblocked logical record (must be at least KBLOK words long).

IEOF

End of file flag

= 0 A logical record has been unblocked

= 1 Indicates an EOF was read

ENTRY POINTS

UBLOK

SPECIAL CONDITIONS

- Will read from any file on one tape.
- Logical record must be an integer multiple of 60 bits long.
- Now set to drop physical records with a parity error.
- See "UZBLOK" for more general case of logical records a fixed number of bits in length.

COMMON BLOCKS

None

I/O

FORTRAN logical unit ITAPE - input tape.

When an EOF is read, the following message is printed:

EOF ON BLOCK READ, PHYS REC=nn, LOG REC=nn .

When a physical record is longer than KMAX, the following message is printed:

ACTUAL PHYSICAL RECORD=nn, MAXIMUM PHYSICAL RECORD=nn .

When IOWAIT returns status of 2 or 3, the following message is printed:

XX TROUBLE ON BLOCK READ TAPE=nn, STATE=nn, WDS=nn, REC=nn, LOG REC=nn . THIS RECORD IS NOT PROCESSED.

10.UBLOK.4

PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Marie Working, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Written by Roy Jenne in February 1967
SPACE REQUIRED	$4014_8 = 2060_{10}$ on 6600; $2351_8 = 1257_{10}$ on 7600.
TIMING	One millisecond per record plus 8 microseconds per word
PORTABILITY	Not portable
REQUIRED RESIDENT ROUTINES	RDTAPE, IOWAIT, OUTPTC, EXIT, Q8QRSD

UZBLOK

SUBROUTINE UZBLOK (ITAPE,MODE,KTYPE,IBUF,KMAX,LREP,NWORDS,KSKIP,KBITS,IEOF)

DIMENSION OF ARGUMENTS LREP(NWORDS),IBUF(KMAX)

LATEST REVISION March 1967

PURPOSE UZBLOK unblocks *fixed length* BCD or binary logical records from *larger* physical records on a "blocked" tape. The logical records do not have to start at the beginning of the physical record and the logical record may be any number of bits long. If the logical records are an integer number of 60-bit words in length, then UZBLOK is more efficient. Logical records may not cross physical record boundaries.

ACCESS CARDS *FORTRAN,S=ULIB,N=UZBLOK
 *COSY

USAGE CALL UZBLOK (ITAPE,MODE,KTYPE,IBUF,KMAX,LREP,NWORDS,KSKIP,KBITS,IEOF)

ARGUMENTS

On Input

ITAPE

Input tape logical unit number.

MODE

Mode for tape read.

= 0 BCD, no character conversion

= 1 BIN, no character conversion

= 2 BCD, conversion from external BCD to display code

= 3 BIN, conversion from external BCD to display code

KTYPE

Type for tape read.

= 0 system

= 2 non-system

IBUF

Input buffer.

KMAX

Maximum physical record size plus one.

NWORDS

Number of full 60-bit words to return for each logical record. Must be large enough to contain KBITS.

KBITS

Number of bits in each logical record.

KSKIP

Number of bits to skip at the start of the physical record. The first logical record starts at bit KSKIP+1.

On Output

LREP

Output buffer for unblocked logical record (must be at least NWORDS words long).

IEOF

End of file flag.

= 0 A logical record has been unblocked.

= 1 Indicates an EOF was read.

ENTRY POINTS

UZBLOK

SPECIAL CONDITIONS

- Will read files from only one tape.
- Logical records may be any fixed number of bits in length. If $60 * NWORDS > KBITS$, the last word of LREP will contain bits not requested. These should be discarded by the user.
- The unblocking from a record stops when there aren't enough bits remaining for another logical record.
- Now set to drop physical records with a parity error.

COMMON BLOCKS

None

I/O

FORTTRAN logical unit ITAPE - input tape.

When an EOF or EOT is read, the following message is printed:

EOF ON BLOCK READ, PHYS REC=nn, LOG REC=nn

I/O
(continued)

When a physical record is longer than KMAX, the following message is printed and program stops:

ACTUAL PHYSICAL RECORD=nn, MAXIMUM PHYSICAL
RECORD=nn .

When IOWAIT returns status of 2 (parity) or 3 (EOT), the following message is printed:

XX TROUBLE ON BLOCK READ TAPE=nn, STATE=nn, WDS=nn,
REC=nn, LOG REC=nn . THIS RECORD NOT PROCESSED.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Roy Jenne, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

Written by Roy Jenne in March 1967.

SPACE REQUIRED

4157₈ = 2159₁₀ on 6600; 2514₈ = 1356₁₀ on 7600

TIMING

One millisecond per record plus 25 microseconds per word.

PORTABILITY

Not portable

REQUIRED RESIDENT
ROUTINES

GBYTES, RDTAPE, IOWAIT, OUTPTC

11

UTILITY ROUTINES

BSEARCH

CHCONV

CONV360

DATE

HOURS

LCKSUM

MOVE

UCHAR

BSEARCH**SUBROUTINE BSEARCH (XBAR,X,N,I,MFLAG)****DIMENSION OF
ARGUMENTS**

X(N)

LATEST REVISION

June 1974

PURPOSE

BSEARCH uses a binary search method to search a table of increasing floating point numbers.

ACCESS CARDS

*FORTRAN,S=ULIB,N=BSEARCH
*COSY

USAGE

CALL BSEARCH (XBAR,X,N,I,MFLAG)

ARGUMENTS**On Input**

XBAR

The argument for which the search is to be performed,
 $X(1) \leq XBAR \leq X(N)$.

On Input
(continued)

X

A real array with dimension N. On input, X must contain the table of floating point values in increasing order to be searched.

N

An integer variable set equal to the number of entries in the array X.

On Output

I

An integer variable set equal to the index of the table entry found equal to XBAR, or to the index of the next smaller entry, if no equal entry is found.

MFLAG

An integer flag set to either 0 or 1.
= 0 if any entry equal to XBAR is found
= 1 if no equal entry is found

ENTRY POINTS

BSEARCH

COMMON BLOCKS

None

I/O

All messages are written using ULIBER.

- If $N < 2$, the subroutine prints the message
N IS LESS THAN 2.
- If XBAR is outside the range of the table, the subroutine prints the message
XBAR IS OUTSIDE RANGE OF TABLE.

- If the elements of the table are not in increasing order, the subroutine prints the message

BSEARCH TABLE NOT IN INCREASING ORDER

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Jo Walsh, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

BSEARCH was originally a subroutine from the Los Alamos Scientific Laboratory, Los Alamos, New Mexico, which was written in February, 1969. It has been revised for the NSSL Library by Jo Walsh, June 1974.

ALGORITHM

The largest integer M , where $2^M \leq N$, is calculated, and XBAR is compared against $X(2^M)$. If $XBAR \neq X(2^M)$, the index for X (i.e., 2^M) is increased or decreased (as determined by the comparison) by the amount 2^{M-1} . This index is in turn changed by the amount 2^{M-2} , and so forth, until an equal condition is found (MFLAG = 0), or if there is no entry in X equal to XBAR, until the first entry smaller than XBAR is found (MFLAG = 1).

ALGORITHM
(continued)

If a given change in the index of X would produce an address outside the table, the index is set to the last element of the table.

SPACE REQUIRED

$214_8 = 140_{10}$

TIMING

To search a table of 1000 elements, BSEARCH takes less than 1 millisecond on the CDC 7600.

PORTABILITY

BSEARCH has been tested on the CDC 6600 and 7600. It is considered to be in ANSI FORTRAN, except that error messages are written using an NCAR system routine, ULIBER.

**REQUIRED RESIDENT
ROUTINES**

ULIBER

SUBROUTINE CHCONV (IA,N,IB,IC,M,ID,IE)

DIMENSION OF
ARGUMENTS

IA(N),IC(M)

LATEST REVISION

April 1968

PURPOSE

CHCONV converts one character set to another character set via a table look-up procedure. Characters in each set are 6 bits in length. There are 10 characters per word.

ACCESS CARDS

*ASCENT,S=ULIB,N=CHCONV
*COSY

USAGE

CALL CHCONV (IA,N,IB,IC,M,ID,IE)

ARGUMENTS

On Input

IA

An input array of N words with 10 characters per word.

N

Dimension of array IA.

IB

Number of characters to skip in the input array before character conversion starts.

ID

Number of consecutive characters to translate after skipping IB characters.

On Output

IC

Output array of converted characters packed 10 characters per word. The array starts with character number IB+1.

M

Dimension of array IC.

IE

Number of invalid characters in input array.

ENTRY POINT

CHCONV

SPECIAL CONDITIONS

The table assembled into CHCONV converts the 64 external BCD characters to corresponding DPC characters. (See the table attached). Invalid characters are set to 53B which is "\$".

COMMON BLOCKS	None
I/O	None
REQUIRED ULIB ROUTINES	None
SPECIALIST	Dennis Joseph, NCAR, Boulder, Colorado 80303
LANGUAGE	ASCENT
B REGISTERS USED	B1, B2, B3, B4, B5, B6, B7
HISTORY	Written in September 1967
SPACE REQUIRED	$161_8 = 113_{10}$
PORTABILITY	Not portable.
REQUIRED RESIDENT ROUTINES	None

A character and symbol table follows.

Character and Symbol Table

<i>External BCD Code</i>	<i>DPC</i>	<i>Symbol</i>
00	53B	\$
01	34	1
02	35	2
03	36	3
04	37	4
05	40	5
06	41	6
07	42	7
10	43	8
11	44	9
12	33	0
13	54	=
14	53	\$
15	53	\$
16	53	\$
17	53	\$
20	55	space
21	50	/
22	23	S
23	24	T
24	25	U
25	26	V
26	27	W
27	30	X
30	31	Y
31	32	Z
32	53	\$
33	56)
34	51	(
35	53	\$
36	53	\$
37	53	\$
40	46	-
41	12	J
42	13	K
43	14	L
44	15	M
45	16	N
46	17	O
47	20	P
50	21	Q
51	22	R
52	53	\$
53	53	\$
54	47	*
55	53	\$

Character and Symbol Table (continued)

<i>External BCD Code</i>	<i>DPC</i>	<i>Symbol</i>
56	53	\$
57	53	\$
60	45	+
61	01	A
62	02	B
63	03	C
64	04	D
65	05	E
66	06	F
67	07	G
70	10	H
71	11	I
72	53	\$
73	57	.
74	52)
75	53	\$
76	53	\$
77	53	\$

PROGRAM CONV360**DIMENSION OF
ARGUMENTS**

None

LATEST REVISION

June 1967

PURPOSE

CONV360 reads IBM 360 (EBCDIC) source cards and punches out CDC BCD source cards. See the attached table for a list of symbols converted by CONV360. The program itself also gives a listing. Punch card and paper limits should be set appropriately.

ACCESS CARDS

*FORTRAN,S=ULIB,N=CONV360
*COSY
*ASCENT,S=ULIB,N=CHANGE
*COSY

ENTRY POINTS

CONV360, CHANGE

SPECIAL CONDITIONS

- The control card, *R80C,(punched starting in column 1 of the card) must immediately precede the input cards.
- A laced card follows the input data. Column 80 must have all rows punched.

COMMON BLOCKS

None

I/O

Input is from FORTRAN logical unit 5 (card reader).
Output is to the card punch, and printer.

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Dennis Joseph, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN, ASCENT

HISTORY

This routine was written in June 1967.

SPACE REQUIRED

$2505_8 = 1394_{10}$

PORTABILITY

Not portable

**REQUIRED RESIDENT
ROUTINES**

RDTAPE

Character and Symbol Table

<i>IBM 360</i> Character Read Into Computer	<i>EBCDIC</i> Card Punches	<i>IBM</i> Symbol	<i>CDC</i> DPC	<i>BCD Card</i> Punches	<i>CDC</i> Symbol
0022B	8-5	1	47B	11-8-4	*
0012B	8-6	=	54B	8-3	=
0006B	7-8	"	47B	11-8-4	*
4000B	12	+	45B	12	+
4042B	12-8-4	<	52B	12-8-4)
1042B	8-4-0	%	51B	0-8-4	(
0102B	8-3	#	54B	8-3	=
0042B	8-4	@	63B	8-6	%
4022B	12-8-5	(51B	0-8-4	(
1102B	8-3-0	,	56B	0-8-3	,
2042B	11-8-4	*	47B	11-8-4	*
2102B	11-8-3	\$	53B	11-8-3	\$
2022B	11-8-5)	52B	12-8-4)
4102B	12-8-3	.	57B	12-8-3	.
2000B	11	-	46B	11	-
4012B	12-8-6	+	45B	12	+
1400B	0-1	/	50B	0-1	/
1001B	0-9	Z	32B	0-9	Z
1002B	0-8	Y	31B	0-8	Y
1004B	0-7	X	30B	0-7	X
1010B	0-6	W	27B	0-6	W
1020B	0-5	V	26B	0-5	V
1040B	0-4	U	25B	0-4	U
1100B	0-3	T	24B	0-3	T
1200B	0-2	S	23B	0-2	S
2001B	11-9	R	22B	11-9	R
2002B	11-8	Q	21B	11-8	Q
2004B	11-7	P	20B	11-7	P
2010B	11-6	O	17B	11-6	O
2020B	11-5	N	16B	11-5	N
2040B	11-4	M	15B	11-4	M
2100B	11-3	L	14B	11-3	L
2200B	11-2	K	13B	11-2	K
2400B	11-1	J	12B	11-1	J
4001B	12-9	I	11B	12-9	I
4002B	12-8	H	10B	12-8	H
4004B	12-7	G	7B	12-7	G
4010B	12-6	F	6B	12-6	F
4020B	12-5	E	5B	12-5	E
4040B	12-4	D	4B	12-4	D
4100B	12-3	C	3B	12-3	C
4200B	12-2	B	2B	12-2	B
4400B	12-1	A	1B	12-1	A
0001B	9	9	44B	9	9

Character and Symbol Table (continued)

<i>IBM 360</i>					
<i>Character Read Into Computer</i>	<i>EBCDIC Card Punches</i>	<i>IBM Symbol</i>	<i>CDC DFC</i>	<i>BCD Card Punches</i>	<i>CDC Symbol</i>
0002B	8	8	43B	8	8
0004B	7	7	42B	7	7
0010B	6	6	41B	6	6
0020B	5	5	40B	5	5
0040B	4	4	37B	4	4
0100B	3	3	36B	3	3
0200B	2	2	35B	2	2
0400B	1	1	34B	1	1
1000B	0	0	33B	0	0
2012B	11-8-6	;	53B	11-8-3	\$

SUBROUTINE DATE (KYR, KMO, KDAY, KHR, KDAWK, KHRS)**DIMENSION OF
ARGUMENTS**

None

LATEST REVISION

January 1966

PURPOSE

Given the number of hours after December 31, 1920, DATE computes date information such as year, month, day of month, hour of day and day of the week (eg. KHRS=1 for 01Z, Jan. 1, 1921). See subroutine HOURS to do the reverse procedure.

ACCESS CARDS

*FORTRAN, S=ULIB, N=DATE
*COSY

USAGE

CALL DATE (KYR, KMO, KDAY, KHR, KDAWK, KHRS)

11.DATE.2

ARGUMENTS

On Input

KHRS

Number of hours after December 31, 1920.

On Output

KYR

Year (4 digits; eg. 1962)

KMO

Month

KDAY

Day of month

KHR

Hour of day

KDAWK

Day of week. 1 through 7 corresponds to Sunday through Saturday.

ENTRY POINTS

DATE

SPECIAL CONDITIONS

- Works for the period 1921 to 2100.
- FORTRAN Standard data statement loads the array MO.

COMMON BLOCKS

None

I/O

None

REQUIRED ULIB ROUTINES	None
SPECIALIST	Roy Jenne, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Written by Roy Jenne at NCAR in January 1966.
SPACE REQUIRED	221 ₈ = 145 ₁₀ on CDC 7600
PORTABILITY	Portable
REQUIRED RESIDENT ROUTINES	None

SUBROUTINE HOURS (NYR,NMO,NDAY,NHRS)DIMENSION OF
ARGUMENTS

None

LATEST REVISION

January 1966

PURPOSE

Given the year, month and day of month, this routine computes the number of hours up to this day after 1920 (eg. NHRS=1 is 01Z 1 JAN, 1921). See subroutine DATE to do the reverse procedure. By making two calls, this routine can be used to find the number of hours between any two dates.

ACCESS CARDS

*FORTRAN,S=ULIB,N=HOURS
*COSY

USAGE

CALL HOURS (NYR,NMO,NDAY,NHRS)

ARGUMENTS

On Input

NYR

Year (4 digits, eg. 1962)

NMO

Month.

NDAY

Day of the month.

On Output

NHRS

Number of hours up to this day after 1920.

ENTRY POINTS

HOURS

SPECIAL CONDITIONS

- Works for the period 1921 to 2100.
- FORTRAN Standard data statement loads the array MO.

COMMON BLOCKS

None

I/O

None

REQUIRED ULIB
ROUTINES

None

SPECIALIST

Roy Jenne, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Written by Roy Jenne at NCAR in January 1966.

SPACE REQUIRED 142₈ = 98₁₀

PORTABILITY Portable

REQUIRED RESIDENT
ROUTINES None

SUBROUTINE MOVE (A,INCA,B,INCB,NWD)**DIMENSION OF
ARGUMENTS**

A(INCA*NWD),B(INCB*NWD)

LATEST REVISION

January 1974

PURPOSE

To provide a rapid in-core transfer of data.

ACCESS CARDS*ASCENT,S=ULIB,N=MOVE
*COSY**USAGE**

CALL MOVE (A,INCA,B,INCB,NWD)

For contiguous words use:

CALL MOVEC (A,B,NWD)

ARGUMENTS

On Input

A

Address of the first word of the source array.

INCA

A array increment. (See Note)

B

Address of the first word of the destination array.

INCB

B array increment.

NWD

Number of words to transfer

On Output

A, INCA, INCB, NWD remain unchanged.

B

Contains those words transferred from A.

NOTES

- NWD words are transferred from A(1), A(1+INCA),... to B(1), B(1+INCB),...

MOVEC assumes INCA = INCB = 1.

- MOVEC allows overlapping arrays by testing, then adjusting the order of the move, i.e., moving from A(NWD), A(NWD-1), ... to B(NWD), B(NWD-1),... when appropriate.

- Either MOVE, MOVEC, or both may be assembled by setting

AOPT = 0 Both.

AOPT = 1 MOVE only.

AOPT = 2 MOVEC only.

The default is AOPT = 0. MOVEC alone is assembled by

```
*ASCENT,S=ULIB,N=MOVE
REPLACE 7
AOPT      EQU      2
*COSY
```

- The assembly variable LSCM controls whether MOVEC will use the LCM (large core memory) block transfer when it is being run on the 7600. To obtain the version using only SCM, use the following.

```
*ASCENT,S=ULIB,N=MOVE
REPLACE 12
LSCM      EQU      1
*COSY
```

- On the 7600 both MOVE and MOVEC with LSCM \neq 0, read ahead three words; hence, care must be taken near the end of one's field length.
- When using SAVE and the default LCM transfer option, the LCM buffer must be requested again after each RESTART. This is accomplished by

```
FARG = SAVEF (ISAV...)
IF (FARG.NE.0.0) CALL MOVEIN
```

The actual LCM request will be issued in the next call to MOVEC.

- If the run is aborted due to there not being enough LCM available, see a systems programmer or SET LSCM=1.

ENTRY POINTS	MOVE, MOVEC, MOVEIN
COMMON BLOCKS	None
I/O	None
REQUIRED ULIB ROUTINES	None
SPECIALIST	Jay W. Chalmers, NCAR, Boulder, Colorado 80303
LANGUAGE	ASCENT
HISTORY	Originally written for the 6600 by G. S. Patterson, Jr. MOVE was rewritten to provide the shorter calling sequence, "in-stack," possibly overlapping moves, and optimized code on either machine.
ALGORITHM	With the exception of the LCM block transfers, NWD is reduced to a multiple of four, the remaining words being moved before entering the main loop.
SPACE REQUIRED	MOVE 42 ₈ MOVEC with LCM 102 ₈ without LCM 42 ₈ Maximum required 144 ₈

PORTABILITY

Not portable

TIMING

Move times can be approximated by a quadratic in $1/M$ where M = number of words moved. For times in μsec , $T(M) = (c_2/M+c_1)/M+c_0$ where the coefficients given below are for:

1. 7600 LCM
2. 7600 SCM contiguous
3. 7600 SCM 3 word increments
4. 7600 SCM 4 word increments (exhibits memory conflicts)
5. 6600 SCM

	C_0	C_1	C_2
1.	0.0582	6.297	-0.0803
2.	0.1203	4.266	0.1800
3.	0.1182	4.982	0.4437
4.	0.2538	4.582	1.362
5.	1.047	24.97	3.867

Comparing the times per word moved for MOVE, a FORTRAN in-line DO loop gives us an indication of whether to use MOVE or not. Timings were made for contiguous words using one, two and four replacements within the loop. Only an integral multiple of the number of replacements were used. Example:

```
DO 1 K = 1,22,2
  B(K) = A(K)
  B(K+1) = A(K+1)
CONTINUE
```

TIMING*(continued)*

This ratio can also be approximated by a quadratic in l/M as above. The break-even points for each case are also given. Use MOVEC when the number of words to be moved is larger than the break-even point.

		C_0	C_1	C_2	<u>Break point</u>
7600 LCM	1.	.1580	16.60	-35.88	18
	2.	.3255	25.74	-92.53	36
	4.	.3789	24.87	-62.93	38
7600 SCM	1.	.3321	10.67	-22.16	13
	2.	.6312	15.11	-53.09	38
	4.	.6855	14.46	-38.62	48
6600 SCM	1.	.5444	9.888	-26.68	17
	2.	.8602	11.05	-38.05	80
	4.	.9604	10.17	-30.89	>150

**REQUIRED RESIDENT
ROUTINE**

None

USAGE*(continued)*

where BEG and END are the first word and the last word, respectively, of the character-string buffer to be scanned (for example, "KARD,KARD(8)").

To retrieve the next character from the current character string, use the statement

```
I=UCHAR (IGO) (3)
```

The character placed in I is in "R1" format (i.e., bits 59-6 are zero, bits 5-0 contain the character). The argument is ignored unless UCHAR finds that the current buffer is exhausted (all characters have been returned), in which case:

if IGO=0, a "STOP" instruction is executed to terminate.

if IGO≠0, it is assumed to have appeared in a previously-executed statement "ASSIGN λ TO IGO" and control is transferred directly to the statement labeled λ.

To facilitate simultaneous scanning of more than one character string, two pairs of entry points are provided: UCHAR1 and UCHAR2; UCHAR3(URA) and UCHAR4(URA).

The statement

```
CALL UCHAR1 (4)
```

causes the current "state" of UCHAR to be saved in a four-word array inside the routine. The subsequent statement

```
CALL UCHAR2 (5)
```

causes the "state" of UCHAR to be restored from that internal four-word array. In between the two calls,

UCHAR0 and UCHAR may be used to scan a different character string. The statements

```
CALL UCHAR3(URA) . . . CALL UCHAR4(URA) (6)
```

perform the same function, except that the four-word user array URA is used to save and restore the state of UCHAR.

ARGUMENTS

BEG and END are the first word and the last word, respectively, of a character-string buffer. Buffer contents are unchanged by these routines.

IGO specifies what UCHAR is to do when an end-of-buffer condition is sensed. IGO=0 => "program stop", IGO≠0 => "GO TO IGO", assuming IGO has appeared in an "ASSIGN" statement.

URA is an array of dimension 4, into which UCHAR3 stores the information necessary to save the current "state" of UCHAR and from which UCHAR4 obtains the information necessary to restore that state.

ENTRY POINTS

UCHAR0, UCHAR, UCHAR1, UCHAR2, UCHAR3, UCHAR4

SPECIAL CONDITIONS

Both of the possible actions taken by UCHAR upon reaching the end-of-buffer are unusual.

COMMON BLOCKS

None

I/O

None

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

D. Kennison, NCAR, Boulder, Colorado 80303

LANGUAGE

ASCENT

HISTORY

Written by R. Helgason about May 1972 as part of the FORTRAN preprocessor "FRED"; carefully optimized by him for the CDC 6600.

SPACE REQUIRED

$75_8 = 61_{10}$

TIMING

None of the six routines mentioned requires (per call) more than 3 μ sec on the 7600 or 15 μ sec on the 6600.

PORTABILITY

Portable to another 6000 or 7000 series machine with whatever minor changes are implied by a different system. Consider linkage changes, differences between ASCENT and some other assembly language, and the "SPECIAL CONDITIONS" mentioned above.

**REQUIRED RESIDENT
ROUTINES**

None

GRAPHICS

AUTOGRAPH

CONREC

CONRECQCK

CONRECSMTH

DASHCHAR

DASHLINE

DASHSMTH

HAFTON

ISOSRF

ISOSRFHR

PWRX

PWRY

PWRZ

SCROLL

SRFACE

SUPMAP

VELVEC

APPENDIX A

APPENDIX B

INTRODUCTION TO GRAPHICS

PHILOSOPHY

At NCAR, the approach has been to have a static machine language system plot package and supplement this with FORTRAN utility routines as needs arise. The Computing Facility program library contains a comprehensive selection of utility routines for generating scientific computer graphics. Generally, these represent the most up-to-date plotting methods consistent with NCAR's graphics philosophy and available hardware.

Because of the nature of scientific computer graphics, algorithms for certain capabilities must be present, and these algorithms must have certain attributes that are often conflicting. For example, some users want the quickest algorithm possible regardless of picture quality while others desire directly publishable plots regardless of the cost. Some users want very simple argument lists, while others need more versatility. Portability of utility routines is also important, and excellent documentation is vital.

CAPABILITIES

The capabilities required for the atmospheric scientist include plotting characters including Greek and Roman alphabets, plotting geographic maps in various projections; automatic generation of plots of curves or families of curves from user supplied arrays; plotting two-dimensional fields via contour maps, half-tone pictures or surfaces with hidden lines removed; displaying two-dimensional

CAPABILITIES
(continued)

velocity fields by drawing velocity vectors or streamlines; and displaying of three-dimensional arrays by drawing contour surfaces or processing two-dimensional slices through the fields. Utility routines for all of these processes have been implemented or created at NCAR.

**PICTURE QUALITY
VS. COST**

There is generally a trade-off between picture quality and algorithm complexity (and resulting code length and execution time). The simplest, smallest, quickest algorithms generally produce the least attractive results; while the most complicated, largest, slowest algorithms generally produce the best looking results. Since no single algorithm can be at both extremes, a family of algorithms is often needed to solve a particular problem. The user can then choose the level of picture quality consistent with his needs by carefully examining the attributes for each package in the table of contents and studying the sample pictures in each write-up.

The construction of dashed lines can illustrate such a hierarchy of routines. The dd80 hardware is capable of producing dashed lines. This hardware, however, fails to produce pleasing results when drawing very short vector segments or segments of varying length. To remedy the problems, a package was written to produce dashed line patterns by software. This package had entry points analogous to the pattern setting and line drawing entries in the system plot package, allowing the user who wished to pay for the slight amount of extra space and time to get better looking results with minimal change to his own program. Subsequently, a version of this package was written that allowed characters to be part of the dashed line pattern and be plotted at the appropriate positions on the line. Finally,

a version was written that automatically fitted a curve through the given points using splines under tension. By selecting one of these four packages, a user may have whatever level of quality he needs.

Applications of these dashed line generators is not restricted to user programs; they can also be used by utility routines. Since the user entry points have the same names in all the software dashed line packages, by calling these routines in higher level packages, the range of picture quality, routine size and speed of the higher level algorithm is attained by merely exchanging dashed line packages. For example, various levels of sophistication in the automatic graphing routines are obtained by using different software dashed line packages with the same driver routines. In the same way, various levels of contouring routines are obtained by using the various dashed line packages with a common driver. Similarly, there are levels of quality available in the character generators, ranging from the dd80 hardware characteristics to very high quality software characters. To prevent name conflicts when two high level packages use the same lower level packages, as in CONREC and AUTOGRAPH both using DASHCHAR, PWRY and DASHCHAR have been added to the binary file and are automatically loaded (only once) when higher level packages reference them. Users wishing to make modifications to these routines may still access them from ULIB. The documentation is written as if the lower level code were present on the file with the higher level code, since loading of the lower level code is transparent to the user.

FLEXIBILITY

The problem of choosing argument lists which possess both simplicity and flexibility is solved by having one of each type. For example, half-tone pictures can be generated by calling an inflexible routine with a short argument list, CALL EZHFTN (Z, M, N), or a flexible routine with a longer argument list, CALL HAFTON (Z, L, M, N, FLO, HI, NLEV, NOPT, NPRM, ISPV, SPVAL). The routine with the short argument list merely calls the one with the long argument list with reasonable values supplied for the additional arguments. The use of clearly identified internal parameters for any value the user might want to alter further increases flexibility and readability.

PORTABILITY

Because the individual scientist is mobile, and because NCAR is becoming the central computer node on the remote job entry network of atmospheric science departments, the portability of the graphics utility routines is extremely important. The portability problem is attacked in two ways: first, a standard FORTRAN is **used** as much as possible, and, second, only a subset of the system plot package entry points are used in the FORTRAN utility routines. Thus, an implementor at a remote site is faced with a total of less than a dozen referenced system plot package entry points for transporting all of NCAR's utility routines that have been brought up to this standard. Appendix 3 contains implementor's write-ups for the portable routine packages.

USAGE*(continued)*

CALL EZMY (Y,L,MANY,NPTS,LABG)

MANY curves drawn.

Y variables plotted as function of first subscript, (i.e.,
Y(I,J), I=1,2,...,NPTS for J=1,2,...,MANY curves).

User supplies graph heading.

CALL EZMXY (X,Y,L,MANY,NPTS,LABG)

MANY curves drawn.

Y variable plotted as function of X variables (i.e.,
Y(I,J), I=1,NPTS, J=1,MANY, as function of either X(I),
I=1,NPTS or X(I,J), I=1,2,...,NPTS,J=1,2,...,MANY curves,
depending on IROW option).

User supplies heading.

CALL ANOTAT (LABX,LABY,IBAC,ISET,NDASH,LDASH)

Resets graph background annotation.

Zero arguments mean no change.

CALL DISPLA (LFRAME,IROW,LTYPE)

Resets frame and log scale options. Also specifies how
plot variables are stored for multiple plots.*Zero arguments mean no change.***ARGUMENTS****On Input For
EXY,EZXY,EZMY,EZMXY****X**Array of independent variables, needed only if explicitly
referenced in call. The reference in the multiple plot
call may be either one X array or MANY X arrays for all
the MANY Y arrays (see IROW option).**Y**Array of variables to be plotted. If a multiple curve
entry, then Y is two dimensional (i.e., normally Y(I,J),
I=1,2,...,NPTS for J=1,2,...,MANY curves, the order of
subscripts can be reversed; see IROW option).**NPTS**

The number of points on each curve.

**On Input For
EXY,EZXY,EZMY,EZMX**

MANY

The number of curves to be plotted from Y array.

L

The length of first subscript of Y array of multiple plot entry. It also applies to the two dimensional X array ($L > NPTS$). When processing the entire array, $L = NPTS$. See Appendix 1 of this chapter for an explanation of using this argument list to process any part of an array.

LABG

Graph heading of 40 characters or less; if less than 40, \$ terminates character string (\$ not plotted, heading automatically centered). If $LABG=0$, then graph is labeled with blanks.

**On Input For
ANOTAT**

LABX

X-axis label of 40 characters or less; if less than 40, \$ terminates character string (\$ not plotted, label automatically centered).

$LABX=0$ means no change from last call. Initial default $LABX=2HX\$$.

LABY

Y-axis label of 40 characters or less; if less than 40, \$ terminates character string (\$ not plotted; label automatically centered).

$LABY=0$ means no change from last call. Initial default $LABY=2HY\$$.

IBAC

Flag to determine what type of plot background to use.

= 0 Leave IBAC as in last call.

= 1 Numerically labeled perimeter background drawn.

= 2 Grid line background drawn with numerically labeled background.

= 3 Half axis background with numerically labeled tic marks.

= 4 No background.

Initial default $IBAC=1$.

**On Input For
ANOTAT**
(continued)

ISET

Flag to control scaling. If ISET is negative, curve will not be drawn. For IABS(ISET)

- = 0 Leave ISET as in last call.
- = 1 AUTOGRAPH does normal call to SET. Rounded limits computed from x and y.
- = 2 Place picture on portion of frame determined by previous call to SET. (First four SET arguments obtained from last SET call.)
- = 3 Use same max and min for x and y as determined by previous call to SET. (Second four SET arguments obtained from last SET call.)

> 4 Options 2 and 3.

Initial default ISET=1.

NDASH

Controls the plot pattern used to draw the plot line (i.e., solid line, labeled line, dashed line, etc.).

= 0 Leave NDASH as in last call.

> 1 Standard 10-bit (or 10-character) dash line patterns assumed in array LDASH. LDASH(I), I=1,...,NDASH. (Generally NDASH=1 or MANY.)

If NDASH=1, LDASH is used for subsequent EZY or EZXY calls.

If NDASH 1, LDASH is used only for subsequent EZMY or EZMYX calls.

< 0 The curves will be solid lines interrupted by character labels. The first curve will be labeled A, the second B, etc.

Initial default NDASH=1.

LDASH

Array of NDASH dash line patterns. A pattern of 0 or 1 is interpreted as a solid pattern. Patterns may be 10 bits or 10 characters as in DASHCHAR.

**On Input For
DISPLA**

LFRAME

Controls the film frame advance.

- < 0 Leave LFRAME as in last call.
- = 1 Frame advance after plotting.
- = 2 No frame advance.
- > 3 Frame advance before plotting.

Initial default LFRAME=1.

IROW

Defines how data is stored in X and Y arrays.

- = 0 Leave IROW as in last call.
- > 0 Data is stored by first subscript, i.e., $Y(I,J)$,
I=1,2,...,NPTS for J=1,2,...,MANY curves.
- < 0 Data stored by second subscript, i.e., $Y(I,J)$,
J=1,2,...,NPTS for I=1,2,...,MANY curves.

IABS(IROW)

- = 1 X array for multiple plots is one dimensional,
i.e., one array of X's for all the many Y's.
- > 1 X array dimensioned the same as Y array for
multiple plots.

Initial default IROW=1.

LTYPE

Controls type of presentation of curve.

- < 0 Leave LTYPE as in last call.
- = 1 Linear X-axis; linear Y-axis.
- = 2 Linear X, log Y.
- = 3 Log X, linear Y.
- > 4 Log x, log Y.

Initial default LTYPE=1.

**On Output For
AUTOGRAPH**

All arguments are unchanged.

NOTE Entry point IDIOT is supported.

ENTRY POINTS EZY, EZXY, EZMY, EZMXY, ANOTAT, DISPLA, LNTH, FTCHLB, SWAPHD, PLOT80, LINRD, LGRD, IDIOT, FRSTD, CURVED, VECTD, LASTD, PWRY, PSYMD, LINED, DASHD

COMMON BLOCKS AUTOGL

I/O Plots curve

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Dan Anderson, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Replaces IDIOT

ALGORITHM Curves are drawn point to point with vector routine.

SPACE REQUIRED About 5300, not including system plot package.

TIMING

For examples following, timing (7600) respectively, is
11 milliseconds, 12 milliseconds, 11 milliseconds.

PORTABILITY

All in FORTRAN but system plot package.

**SYSTEM PLOTTING
ROUTINES USED**

SET, OPTION, VECTOR, GETSET, DASHLN, LABMOD, CURVE, MXY,
PWRT, FRAME, FRSTPT, GRIDALL

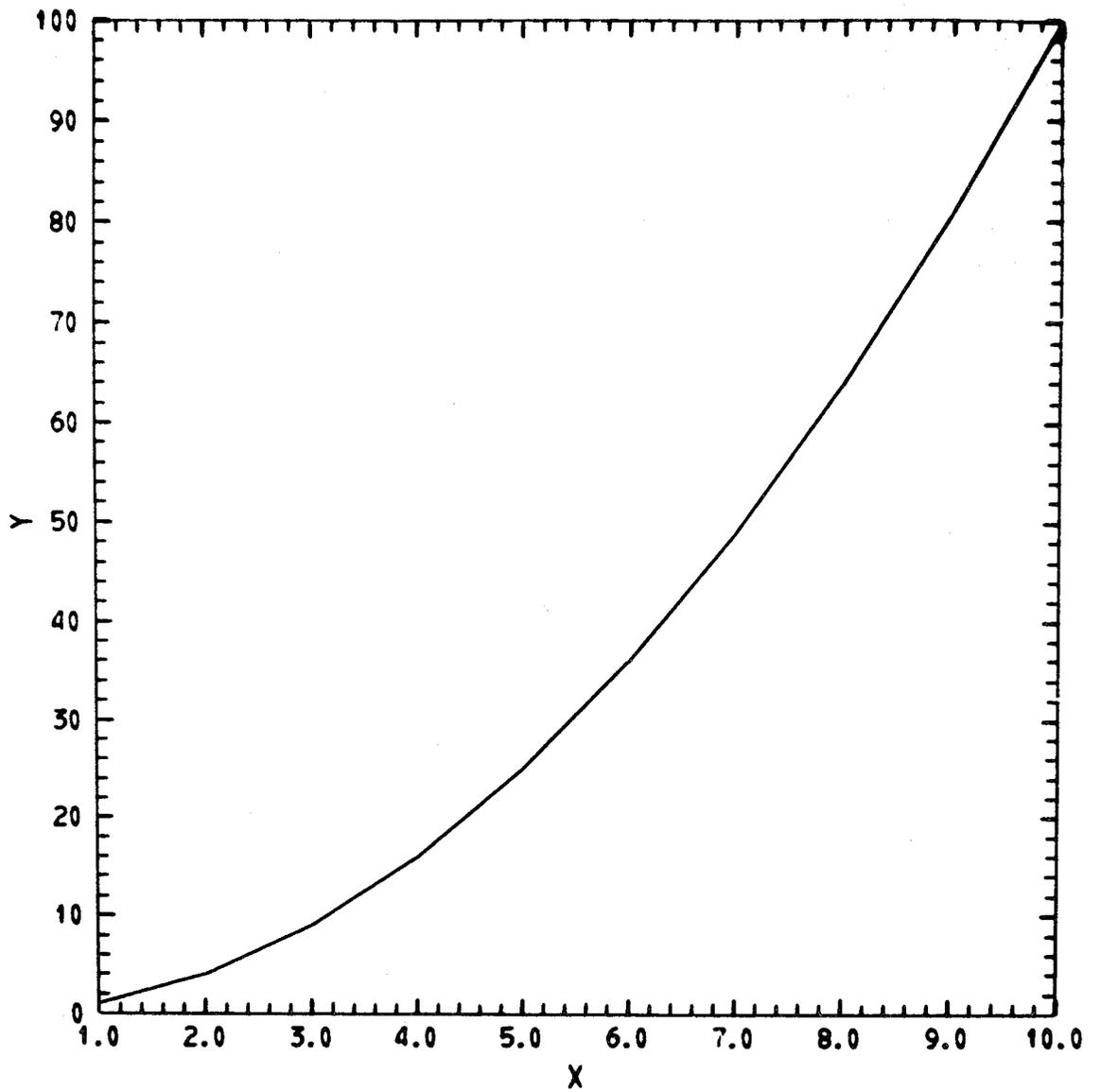
**REQUIRED RESIDENT
ROUTINES**

ALOG10, ATAN2

Please see the following examples.

Example 1

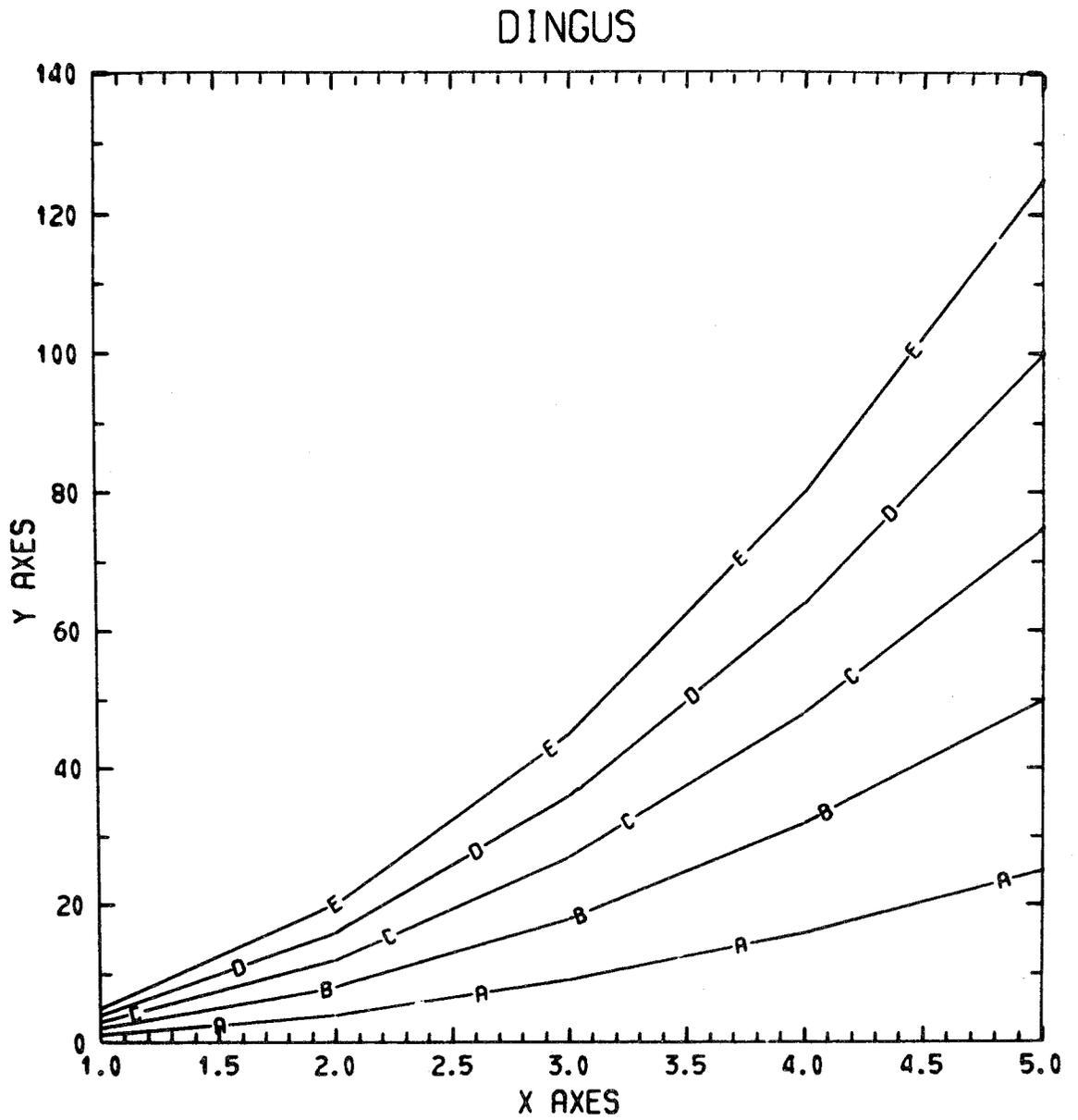
Call EZY (Y,10,0)



Example 2

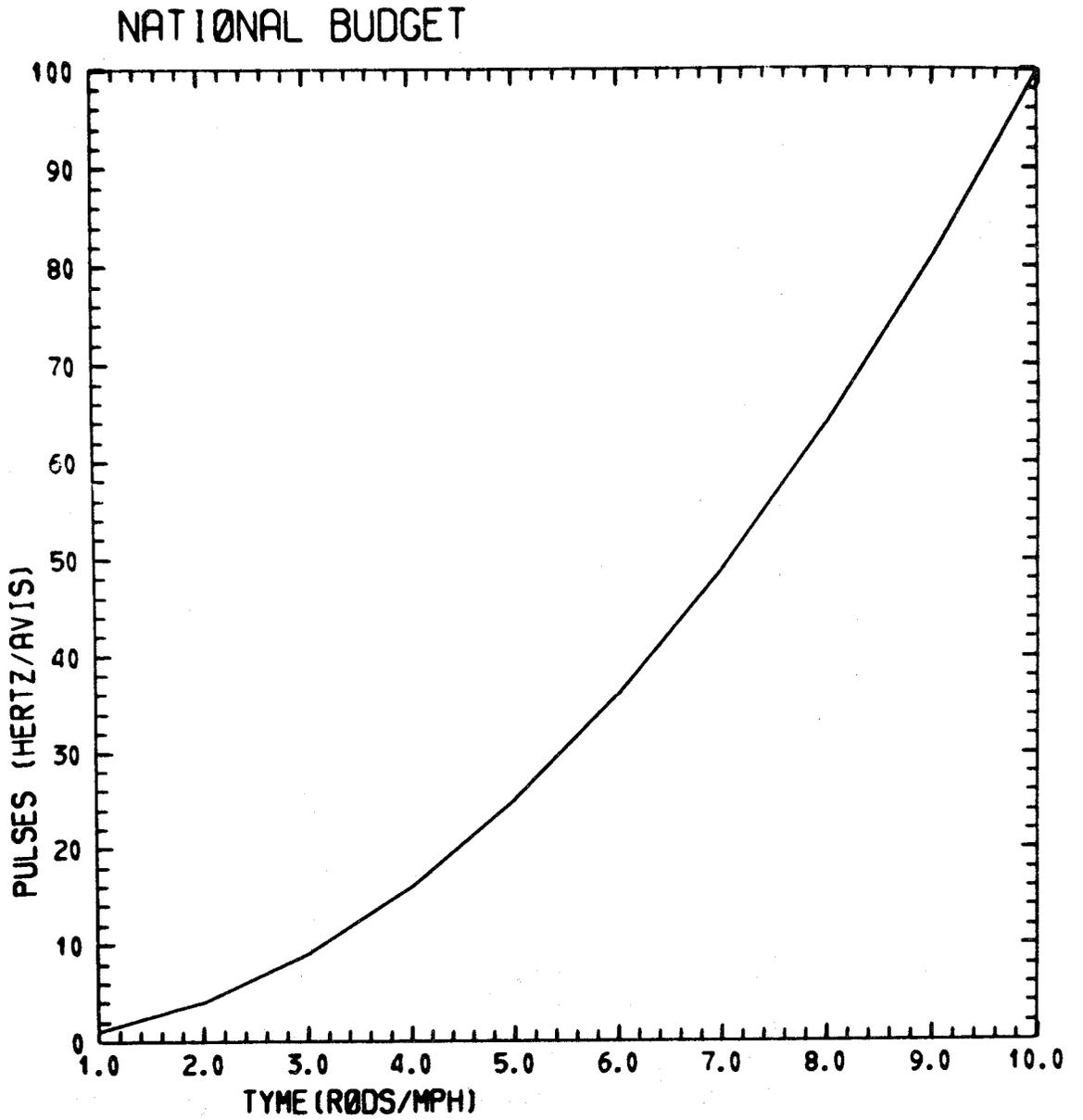
CALL ANOTAT (7HX AXES, 7HY AXES,0,0,0,0)

CALL EZMY (Y,10,5,5,7HDINGUS\$)



Example 3

CALL IDIOT (X,Y,10,1,1777B,LABX,LABY,LTTT,0)



CONREC STANDARD**SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)**

**DIMENSION OF
ARGUMENTS** Z(L,N)

LATEST REVISION June, 1973

PURPOSE CONREC draws a contour map from data stored in a rectangular array. Contour lines are labelled.

ACCESS CARDS *FORTRAN,S=ULIB,N=CONREC
 *COSY

USAGE If the following assumptions are met, use

CALL EZCNTR (Z,M,N)

Assumptions:

All of the array is to be contoured,
Contour levels are picked internally,
Contouring routine picks scale factors,
Highs and Lows are marked,
Negative lines are drawn with a dashed line pattern,
EZCNTR calls FRAME after drawing the contour map.

If these assumptions are not met, use

CALL CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

ARGUMENTS

On Input
for EZCNTR

Z
M by N array to be contoured.

M
First dimension of Z.

N
Second dimension of Z.

On Output
for EZCNTR

All arguments are unchanged.

On Input
for CONREC

Z
The (origin of the) array to be contoured. Z is L by N.

L
The first dimension of Z in the calling program.

M
The number of data values to be contoured in the x-direction (the first subscript direction). When plotting an entire array, $L = M$. See Appendix 1 of the Graphics Chapter for an explanation of using this argument list to process any part of an array.

N
The number of data values to be contoured in the y-direction (the second subscript direction).

FLO
The value of the lowest contour level. If $FLO = HI = 0.$, a value rounded up from the minimum Z is generated by CONREC.

HI
The value of the highest contour level. If $HI = FLO = 0.$, a value rounded down from the maximum Z is generated by CONREC.

FINC

- > 0 Increment between contour levels.
- = 0 A value which produces between 10 and 30 contour levels at nice values is generated by CONREC.
- < 0 The number of levels generated is ABS(FINC).

NSET

Flag to control scaling.

- = 0 CONREC calls SET to properly scale the plotting instructions to the standard configuration. PERIM is called and puts tick marks corresponding to the data points.
- > 0 CONREC assumes that SET has been called by the user in such a way as to properly scale the plotting instructions generated by CONREC. PERIM is not called.
- < 0 CONREC calls SET in such a way as to place the contour plot within the limits of the user's last SET call. PERIM is not called.

NHI

Flag to control extra information on the contour plot.

- = 0 Highs and lows are marked with an H or L as appropriate, and the value of the high or low is plotted under the symbol.
- > 0 The values in each Z are plotted at each Z point, with the center indicating the data point location.
- < 0 Neither of the above are done.

NDOT

A 10-bit constant designating the desired dashed line pattern the contour lines in the plot if |NDOT| is set to 0, 1, or 1777B, solid lines are drawn.

- > 0 Pattern is used for all lines.
- < 0 Pattern is used for negative-valued contour lines.

See DASHD comments and *NCAR Library Routines Manual*, page 4.26.

NOTE	See the listed reference for directions for the most common modifications								
	<table> <tr> <td>Comparison of contouring</td> <td>Contouring write-up</td> </tr> <tr> <td>Non-uniform contour levels</td> <td>Comments in CLSET</td> </tr> <tr> <td>Transformations of contour lines</td> <td>Appendix 2 to Graphics Chapter</td> </tr> <tr> <td>Internal parameters such as label sizes. Unknown data points, drawing placement, etc.</td> <td>Internal parameters (below)</td> </tr> </table>	Comparison of contouring	Contouring write-up	Non-uniform contour levels	Comments in CLSET	Transformations of contour lines	Appendix 2 to Graphics Chapter	Internal parameters such as label sizes. Unknown data points, drawing placement, etc.	Internal parameters (below)
Comparison of contouring	Contouring write-up								
Non-uniform contour levels	Comments in CLSET								
Transformations of contour lines	Appendix 2 to Graphics Chapter								
Internal parameters such as label sizes. Unknown data points, drawing placement, etc.	Internal parameters (below)								
ENTRY POINTS	CONREC, CLGEN, REORD, ENCD, STLINE, DRLINE, MINMAX, PNTVAL, DASHD, CURVED, FRSTD, VECTD, LINED, PSYMD, PWRY, CALCNT, EZCNTR								
COMMON BLOCKS	<table> <tr> <td>CONREL</td> <td>2</td> </tr> <tr> <td>CONRE2</td> <td>1014B</td> </tr> <tr> <td>DASHD1</td> <td>147B</td> </tr> <tr> <td>DASHD2</td> <td>1</td> </tr> </table>	CONREL	2	CONRE2	1014B	DASHD1	147B	DASHD2	1
CONREL	2								
CONRE2	1014B								
DASHD1	147B								
DASHD2	1								
I/O	Plots contour map.								
PRECISION	Single								
REQUIRED ULIB ROUTINES	None								
SPECIALIST	Thomas Wright, NCAR, Boulder, Colorado 80302								
LANGUAGE	FORTRAN								
HISTORY	Replaces old contouring package called CALCNT at NCAR.								
ALGORITHM	Each line followed to completion, points along line found on boundaries of the (rectangular) cells, points along line connected by straight lines using the software dashed line package, which labels the lines.								

SPACE REQUIRED About 7000 octal not including system plot package.

TIMING Varies widely with size and smoothness of Z. Average smoothness 30 by 30 array takes less than two seconds on CDC 6600.

PORTABILITY An implementors write-up is available.

PLOTTING ROUTINES USED SET, GETSET, PERIM, FRSTPT, VECTOR, MXY, CURVE, OPTION (to set intensity), and DASHLN

REQUIRED RESIDENT ROUTINES ALOG10, SQRT, ATAN2, SIN, COS

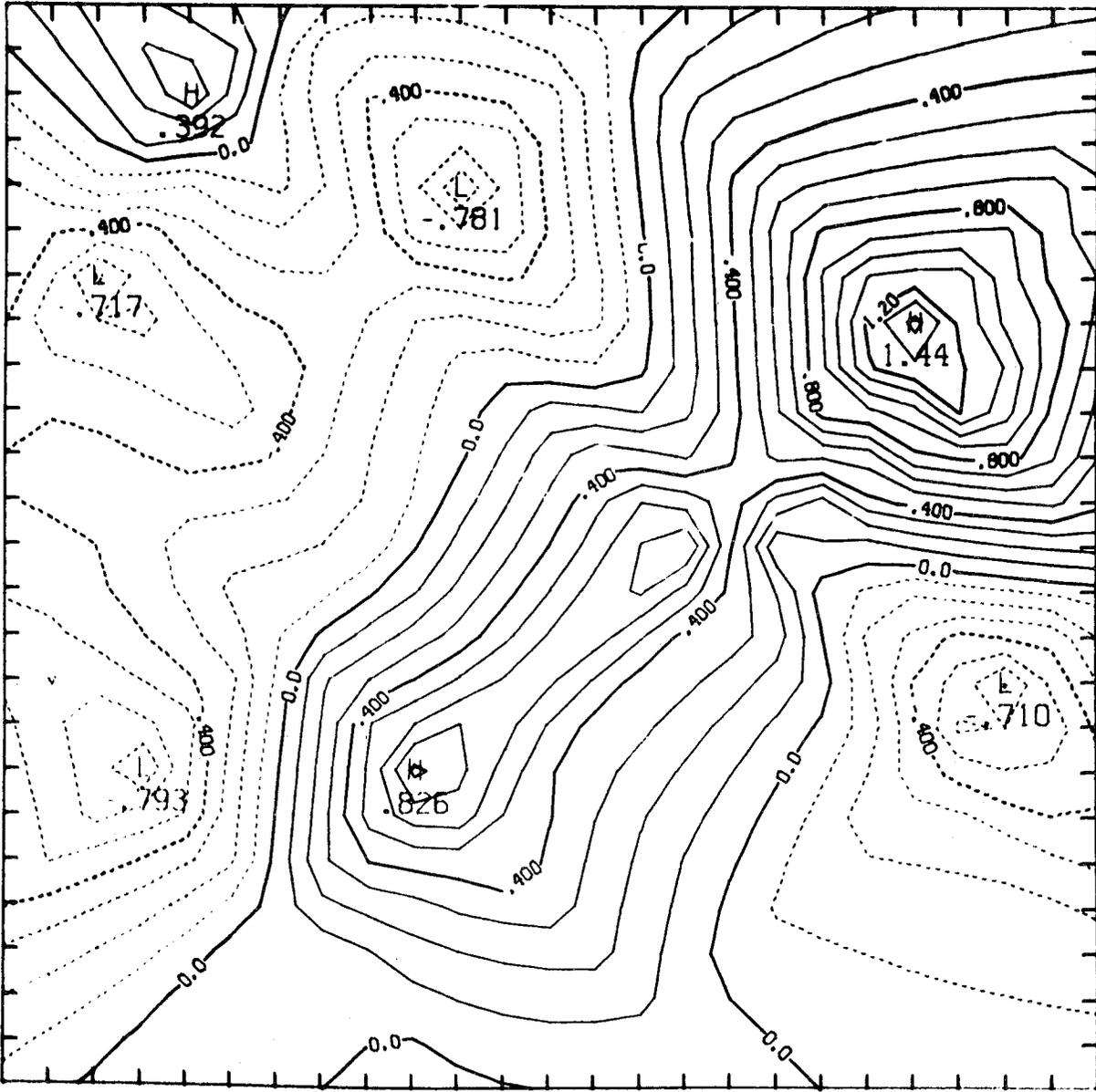
INTERNAL PARAMETERS

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
SIZEL	1.5	Size of line labels.
SIZEM	2.5	Size of labels for minimums and maximums.
SIZEP	1.0	Size of labels for data point values.
NLA	16	Approximate number of contour levels when internally generated.
NLM	40	Maximum number of contour levels. If this is to be increased, the dimensions of CL and IWORK must be increased by the same amount in CONREC.
XLT	.05	Left hand edge of the plot (0.0 = left edge of frame).
YBT	.05	Bottom edge of the plot (0.0 = bottom of frame. 1.0 = top of frame).
SIDE	0.9	Length of longer edge of plot (see also EXT).
NREP	6	Number of repetitions of the dash pattern between line labels.
NCRT	4	Number of CRT units per element (bit) in the dash pattern.
ILAB	1	Flag to control the drawing of line labels. <ul style="list-style-type: none"> ● ILAB non-zero means label the lines. ● ILAB=0 means do not label the lines.

INTERNAL PARAMETERS
(continued)

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
NULBLL	3	Number of unlabeled lines between labeled lines. For example, when NULBILL = 3, every fourth level is labeled.
IOFFD	0	Flag to control normalization of label numbers. <ul style="list-style-type: none"> • IOFFD = 0 means include decimal point when possible (do not normalize unless required). • IOFFD non-zero means normalize all label numbers and output a scale factor in the message below the graph.
EXT	.25	Lengths of the sides of the plot are proportional to M and N (when CONREC calls SET) except in extreme cases, namely, when the $\text{MIN}(M,N)/\text{MAX}(M,N)$ is less than EXT. Then CONREC produces a square plot.
IOFFP	0	Flag to control special value feature. <ul style="list-style-type: none"> • IOFFP = 0 means special value feature not in use. • IOFFP non-zero means special value feature in use. SPVAL is set to the special value. Contour lines will then be omitted from any cell with any corner equal to the special value.
SPVAL	0.	Contains the special value when IOFFP is non-zero.
IOFFM	0	Flag to control message below plot. <ul style="list-style-type: none"> • IOFFM = 0 means message is plotted. • IOFFM non-zero means the message is not plotted.
ISOLID	1777B	Dash pattern for non-negative lines.

A sample picture follows:



CONTOUR FROM -.7000 TO 1.4000 CONTOUR INTERVAL OF .1000 PT(3,3) = -.11191

A QUICKENED VERSION OF THE CONTOUR PACKAGE, CONRECDIMENSION OF
ARGUMENTS

Z(L,N)

LATEST REVISION

December 1973

PURPOSE

Draws a contour map from data stored in a rectangular array.
No line labels

ACCESS CARDS

*FORTRAN,S=ULIB,N=CONRECQCK
*COSY

USAGE

See CONREC write-up in *NCAR Scientific Subroutine Library Manual*.

ARGUMENTS

See CONREC write-up in *NCAR Scientific Subroutine Library Manual*.

Note: NDOT is a 10-bit constant designating the desired dashed line pattern for all the contour lines in the plot. If NDOT is set to 0, 1, or 1777B, solid lines are drawn.

ARGUMENTS
(continued)

See DASHLN comments in *NCAR Scientific Subroutine Library Manual*.

ENTRY POINTS

CONREC, CLCEN, ENCD, MINMAX, QUICK, CALCNT, EZCNTR

COMMON BLOCKS

CONREL 2 words

I/O

Plots contour map

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Tom Wright and Bonnie Gacnik, NCAR, Boulder, Colorado 80303.

LANGUAGE

FORTRAN

HISTORY

A faster version of CONREC *without* line labelling capabilities.

ALGORITHM

The grid space is divided into $(M-1)*(N-1)$ cells. Each cell is processed in turn, drawing all contour lines found in a particular cell, until the entire rectangular space is contoured.

SPACE REQUIRED

About 3000 octal not including system plot package.

TIMING

Varies widely with size and smoothness of Z. Takes 1/2 as long as CONREC (Standard). An implementor's write-up is available. The sample figure took .06 seconds on the 7600.

PLOTTING ROUTINES USED

SET, GETSET, PERIM, FRSTPT, MXY, LINE, DASHLN, PWRT, FRAME

REQUIRED RESIDENT ROUTINES

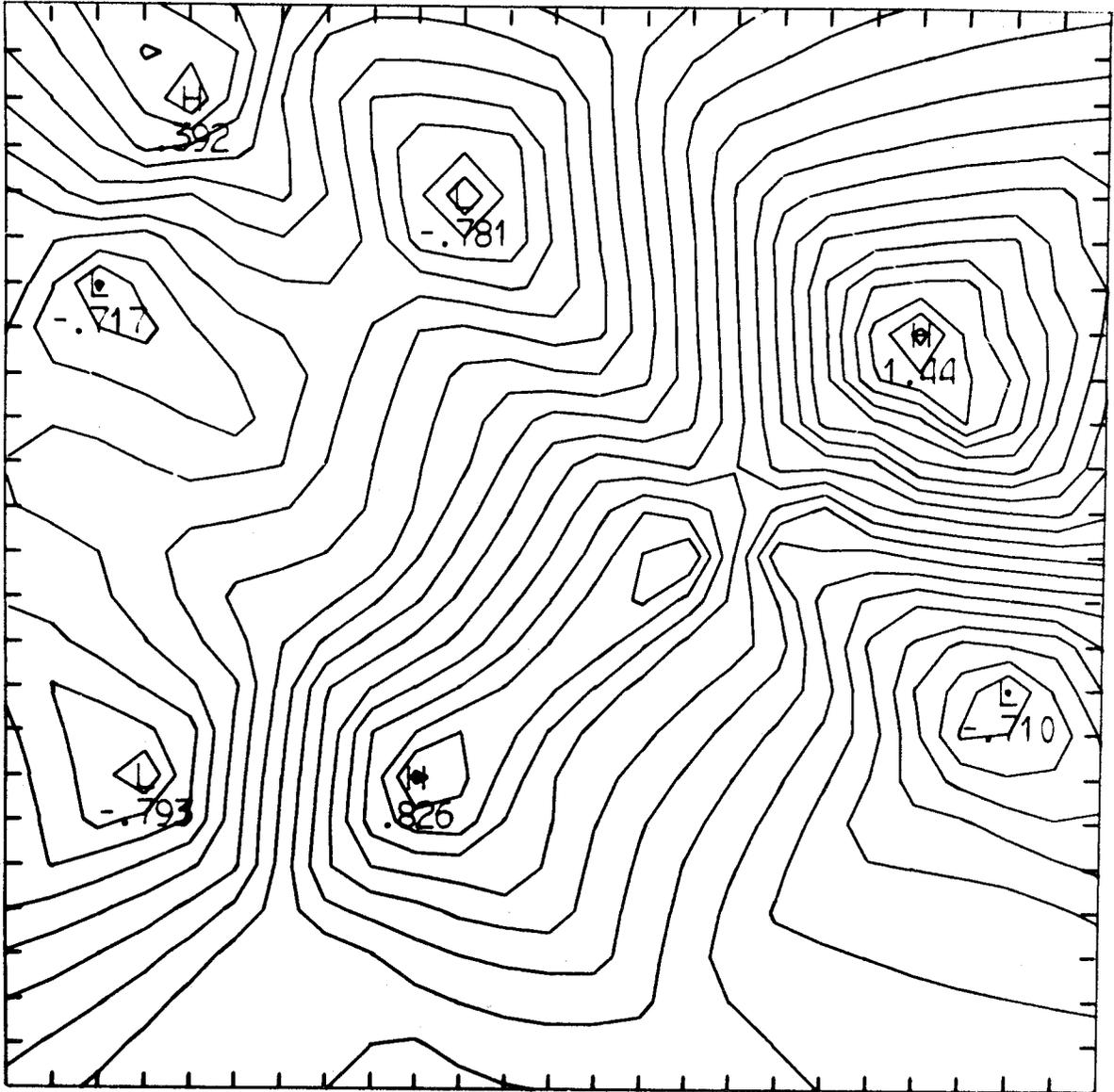
ALOG10

INTERNAL PARAMETERS

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
ISIZEM	2	Size of labels for minimums and maximums found in subroutine PWRT.
ISIZEP	1	Size of labels for data point values found in subroutine PWRT.

For explanations of NLA, SLT, YBT, SIDE, IDFFD, EXT, IDFFP, SPVAL, IDFFM, ISOLID, see CONREC write-up, *NCAR Scientific Subroutine Library Manual*.

A sample picture follows:



CONTOUR FROM -.70000 TO 1.4000 CONTOUR INTERVAL OF .10000 PT(5,3)= -.11191

CONRECSMTH**A SMOOTHING VERSION OF THE CONTOUR PACKAGE, CONREC**

DIMENSION OF ARGUMENTS Z(L,N)

LATEST REVISION November 1973

PURPOSE CONRECSMTH draws a contour map from data stored in a rectangular array. Contour lines are labelled and smoothed.

ACCESS CARDS *FORTRAN,S=ULIB,N=CONRECSMTH
 *COSY

USAGE Same as CONREC standard, that is,

 CALL EZCNTR (Z,M,N)

 or

 CALL CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

ARGUMENTS See CONREC standard.

NOTE • CONRECSMTH is CONREC with the software dashed line package with character capability (DASHCHAR) replaced by the software dashed line package with character capability and smoothing (DASHSMTH).

NOTE
(continued)

- Mods to CONREC STANDARD can also be used on this file without change.
- With the following exceptions, everything is the same as CONREC STANDARD writeup.

ENTRY POINTS

CONREC, CLGEN, REORD, ENCD, STLINE, DRLINE, DASHD, CRVED, PSYMD, PWRY, CURVED, CALCNT, FRSTD, KURV1, KURV2S, MINMAX, PNTVAL, VECT, LINED, VECTD, LASTD, FRST

SPECIALIST

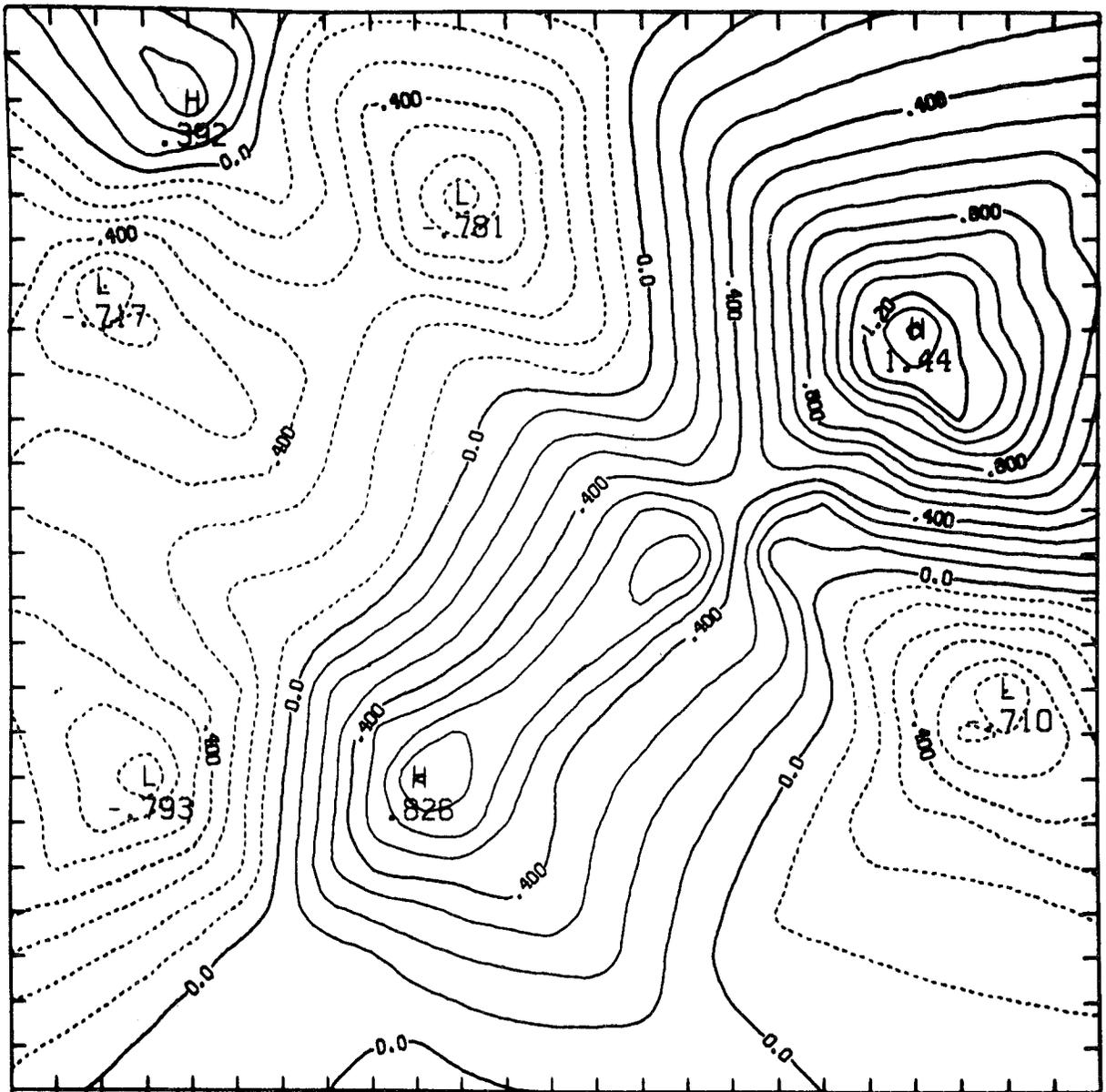
Thomas Wright or Bonnie Gacnik, NCAR, Boulder, Colorado 80302

TIMING

About 1.6 times as long as CONREC STANDARD.

SPACE REQUIRED

11733₈



CONTOUR FROM $-.70000$ TO 1.4000 CONTOUR INTERVAL OF $.10000$ PT(3,3) = $-.11191$

SOFTWARE DASHED LINE PACKAGE WITH CHARACTER CAPABILITY

LATEST REVISION November 1972

PURPOSE DASHCHAR is a software dashed line package. The dd80 hardware dashed line generator fails to produce pleasing results when drawing very short vector segments or vector segments of varying length. DASHCHAR does not have this problem. In addition, this package can put characters at intervals in the line to label the line.

ACCESS CARDS *FORTRAN,S=ULIB,N=DASHCHAR
 *COSY

USAGE First,

CALL DASHD (IPAT,NC,JCRT,ISIZE)

then, call any of the following:

CALL CURVED (X,Y,N)
CALL FRSTD (X,Y)
CALL VECTD (X,Y)
CALL LINED (XA,YA,XB,YB)
CALL PSYMD (X,Y,IDPC,IS,IC,IPEN)

At any time, the character drawing routine may be called

CALL PWRY (X,Y,ID,N,ISIZE,ITHETA,ICENT)

PWRY has some capabilities not found in PWRT of the system plot package.

ARGUMENTS

Except for DASHD and PWRY, all arguments match those in the corresponding routine in the system plot package. See NCAR *Library Routines Manual*, Chapter 4 for explanation of the arguments.

<u>DASHCHAR PACKAGE</u>	<u>SYSTEM PLOT PACKAGE</u>	<u>NCAR LRM</u>
CURVED (X,Y,N)	CURVE (X,Y,N)	4.14
FRSTD (X,Y)	FRSTPT (X,Y)	4.12
VECTD (X,Y)	VECTOR (X,Y)	4.12
LINED (XA,YA,XB,YB)	LINE (XA,YA,XB,YB)	4.14
PSYMD (X,Y,IDPC,IS,IC,IPEN)	PSYM (X,Y,IDPC,IS,IC,IPEN)	4.13

Wherever the system plot package allows either fixed or floating point arguments, the software dashed line package does also.

**On Input
for DASHD****IPAT**

A dash line pattern that may be defined in one of two ways:

- If IPAT is an integer in the range 0-1777B the last 3 arguments are neither needed nor used. These 10 bits are treated as a code for solid or gap with a one bit=3 CRT units solid, and a zero bit as 3 CRT units of gap. Thus the pattern 1470B = 1100111000 means 6 on, 6 off, 9 on, 9 off, etc.
- If IPAT is not of the form above, it is assumed to be a Hollerith string (NC) characters long. NC may be any number, although about 60 seems to be a practical limit. A solid is indicated by a \$. A gap is indicated by a quote (' on the keypunch, ≠ when printed). Blanks are ignored. Any other Hollerith characters become part of the pattern with a hole of appropriate length being left in the line for the character string. No single Hollerith string to be printed as such may be longer than 9 characters.

NC

The number of characters in IPAT.

JCRT

The length in CRT units per \$ or quote.

JSIZE

Is the size of the PWRY character. JSIZE may be floating or integer.

- If floating, it is a multiplication factor of a 6 CRT character width.
- If integer and between 0 and 3 the multiply factor is chosen as in PWRT, i.e., 1., 1.5, 2., and 3. times the 6 CRT width.
- If integer and greater than 3, it is the character width in CRT units.

On Input
for PWRY

X,Y

Positioning coordinates for the characters to be drawn. These can be integers in the range 1 through 1024 or floating point numbers, in which case they are scaled according to the most recent SET call. Also see ICNT.

ID

Characters to be drawn.

N

The number of characters in ID.

ISIZE

Size of the character. ISIZE may be floating or integer.

- If floating, it is a multiplication factor of a 6 CRT character width.
- If integer between 0 and 3, the factor is chosen as in PWRT (1., 1.5, 2., 3. times the 6 CRT width).
- If integer greater than 3, it is the character width in CRT units.

ITHETA

Angle counter clockwise from X axis that the characters are plotted at.

- If integer, $ANGLE = ITHETA * 90$ degrees.
- If floating, $ANGLE = ITHETA$ radians.

On Input
for PWRY
(continued)

ICNT

Centering option.

- If ICNT = 0, then (X,Y) is the center of the first character.
- If ICNT non-zero, then (X,Y) is the center of the entire string.

On Output

All arguments are unchanged for all routines.

NOTE

- The DASHD pattern is not cleared to SOLID by a PWRT or PSYM call as is the hardware dash line generator. It can only be changed by another DASHD call.
- DASHD calls DASHLN (the plot package hardware dash line routine) with an all solid pattern. The other entries *assume* that the hardware dash line is therefore set to solid.
- Calls to the regular plotting routines may be made at any time and do not affect this package, i.e., they are completely independent, except that one must call FRSTPT or FRSTD appropriately when switching from one package to the other and using VECTOR or VECTD.
- When using FRSTD and VECTD, LASTD may be called (no arguments needed). If the dashed line package was leaving a space for characters but the line ended before there was enough room for the characters, the space left will be filled in if LASTD is called.

ENTRY POINTS

DASHD, CURVED, FRSTD, VECTD, LINED, PSYMD, LASTD, PWRY

COMMON BLOCKS

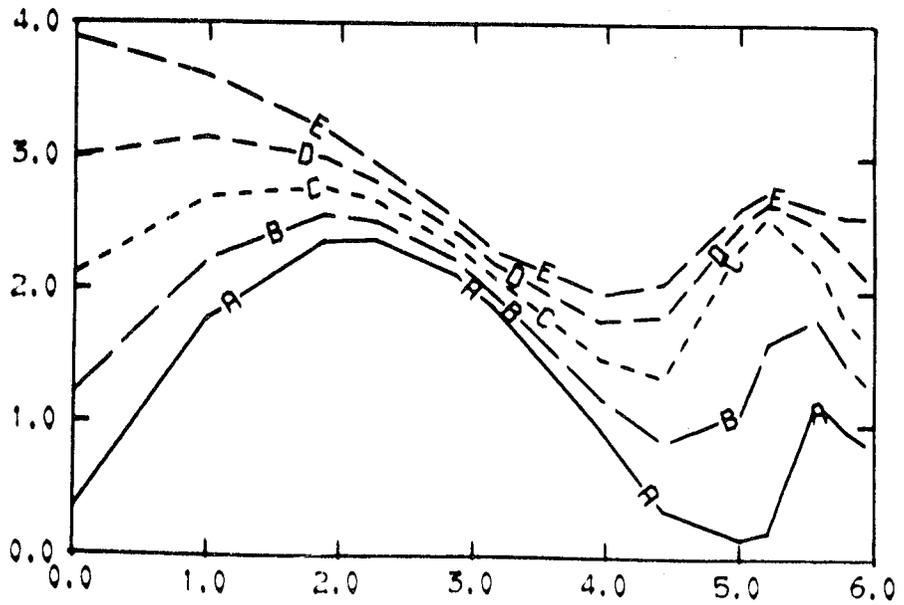
DASHD1 147₈
DASHD2 1

I/O	Plots lines.
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Thomas Wright, NCAR, Boulder, Colorado 80302
LANGUAGE	FORTRAN
HISTORY	Written in 1969, standardized November 1972.
ALGORITHM	Position in dash pattern is remembered as points are processed. Distance traversed in CRT space is used to determine whether to draw segments, parts of segments, characters, or nothing.
SPACE REQUIRED	About 1100 ₈
ACCURACY	±.5 CRT units per call. There is no cumulative error.
TIMING	For solid or blank lines, there is almost no overhead. For dashed lines, about 4 times as long as the system plot package (but the line drawing software in the system plot package is so fast that the increase will not be noticeable in most programs). For lines with characters, not much longer.
PORTABILITY	An implementor's writeup is available.
PLOTTING ROUTINES USED	DASHLN, CURVE, FRSTPT, MXY, VECTOR, PSYM, OPTION

REQUIRED RESIDENT ROUTINES SQRT, ATAN2, SIN, COS

INTERNAL PARAMETERS

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
KCRT	3	Number of CRT units per element (bit) in the dash pattern when $\leq 1777B$. Thus, the pattern is repeated every $ICRT*10$ CRT units.
FPART	1.	Multiplying factor for first solid line segment. This can be used to off-set labels. For example, if $FPART = .5$, the first solid line segment is only one-half as long as it is the rest of the time. This moves all labels on this line towards the beginning, which reduces the probability of the label being written on top of a label of a nearby line drawn with $FPART = 1$.
IGP	9	Flag to control the space left for characters. = 9 gap is left. = 0 no gap is left.



C X,Q,R,S,T,AND U ARE PREVIOUSLY DEFINED ONE DIMENSIONAL
C ARRAYS

C SET AND PERIML HAVE BEEN CALLED

CALL DASHD(20H\$A,20,10,2.)

CALL CURVED(X,Q,N)

CALL DASHD(20H\$\$\$\$'\$\$\$\$'\$\$\$\$'\$\$\$\$'\$\$\$\$B,20,10,2.)

CALL CURVED(X,R,N)

CALL DASHD(20H\$'\$'\$'\$'\$'\$'\$'\$'\$'\$'\$C,20,10,2.)

CALL CURVED(X,S,N)

CALL DASHD(20H\$\$'\$\$'\$\$'\$\$'\$\$'\$\$D ,20,10,2.)

CALL CURVED(X,T,N)

CALL DASHD(20H\$\$\$'\$\$\$'\$\$\$'\$\$\$'\$\$\$E,20,10,2.)

CALL CURVED(X,U,N)

CALL FRAME

ARGUMENTS

On Input

All arguments match those in the corresponding routine in the system plot package. See NCAR *Library Routines Manual*, Chapter 4 for explanation of the arguments.

<u>DASHLINE PACKAGE</u>	<u>SYSTEM PLOT PACKAGE</u>	<u>NCAR LRM</u>
DASHD (IPAT)	DASHLN (IPAT)	4.26
CURVED (X,Y,N)	CURVE (X,Y,N)	4.14
FRSTD (X,Y)	FRSTPT (X,Y)	4.12
VECTD (X,Y)	VECTOR (X,Y)	4.12
LINED (XA,YA,XB,YB)	LINE (XA,YA,XB,YB)	4.14
PSYMD (X,Y,IDPC,IS,IC,IPEN)	PSYM (X,Y,IDPC,IS,IC,IPEN)	4.13

Wherever the system plot package allows either fixed or floating point arguments, the software dash line package does also.

On Output

All arguments are unchanged.

NOTE

- The DASHD pattern is not cleared to SOLID by a PWRT or PSYM call as is the hardware dash line generator. It can only be changed by another DASHD call.
- DASHD calls DASHLN (the plot package hardware dash line routine) with an all solid pattern. The other entries *assume* that the hardware dash line is therefore set to solid.
- Calls to the regular plotting routines may be made at any time and do not affect this package, i.e., they are completely independent, except that one must call FRSTPT or FRSTD appropriately when switching from one package to the other and using VECTOR or VECTD.

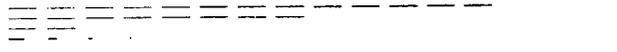
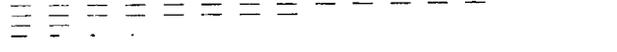
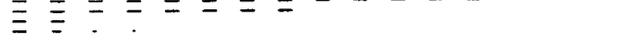
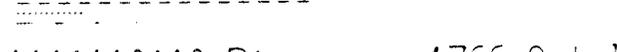
ENTRY POINTS	DASHD, CURVED, FRSTD, VECTD, LINED, PSYMD
COMMON BLOCKS	DSHD 14B
I/O	Plots lines.
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Thomas Wright, NCAR, Boulder, Colorado 80302
LANGUAGE	FORTRAN
HISTORY	Written in 1969, standardized November 1972.
ALGORITHM	Position in dash pattern is remembered as points are processed. Distance traversed in CRT space is used to determine whether to draw segments, parts of segments, or nothing.
SPACE REQUIRED	513B
ACCURACY	± 0.5 CRT units per call. There is no cumulative error.
TIMING	For solid or blank lines, there is almost no overhead. For dashed lines, about 4 times as long as the system plot package (but the line drawing software in the system plot package is so fast that the increase will not be noticeable in most programs).
PORTABILITY	Reasonably portable.

PLOTTING ROUTINES USED DASHLN, CURVE, FRSTPT, MXY, VECTOR, PSYM

REQUIRED RESIDENT ROUTINES SQRT

INTERNAL PARAMETERS	<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
	ICRT	3	Number of CRT units per element (bit) in the dash pattern. Thus, the pattern is repeated every ICRT*10 CRT units.

A comparison of hardware and software dashed line patterns is illustrated (note that the lengths of the dashes with the software dashed lines is the same regardless of the line length):

Hardware Dashed Line Patterns	Software Dashed Line Patterns
1111111100 Binary or 1774 Octal 	1111111100 Binary or 1774 Octal 
1111110000 Binary or 1760 Octal 	1111110000 Binary or 1760 Octal 
1111000000 Binary or 1700 Octal 	1111000000 Binary or 1700 Octal 
1100000000 Binary or 1400 Octal 	1100000000 Binary or 1400 Octal 
1100011000 Binary or 1430 Octal 	1100011000 Binary or 1430 Octal 
1111011110 Binary or 1736 Octal 	1111011110 Binary or 1736 Octal 
1111110110 Binary or 1766 Octal 	1111110110 Binary or 1766 Octal 

SOFTWARE DASHED LINE PACKAGE WITH CHARACTER CAPABILITY AND SMOOTHING

LATEST REVISION October 1973

PURPOSE DASHSMTH is a software dashed line package with smoothing capabilities. DASHSMTH is DASHCHAR with smoothing features added.

ACCESS CARDS *FORTRAN,S=ULIB,N=DASHSMTH
 *COSY

USAGE First,

 CALL DASHD (IPAT,NC,ICRT,ISIZE)

then, call the following:

CALL CURVED (X,Y,N)

or,

CALL FRSTD (X,Y)
CALL VECTD (X,Y)
CALL LASTD

FRSTD processes the first point of a line. VECTD is called for the remaining points of a line. LASTD is called only after the last point of a line has been processed in VECTD.

USAGE*(continued)*

The following may also be called, but no smoothing will result:

```
CALL LINED (XA,YA,XB,YB)
CALL PSYMD (X,Y,IDPC,IS,IC,IPEN)
```

ARGUMENTS

All arguments match those in the corresponding routines in DASHCHAR.

NOTE

When using FRSTD and VECTD, LASTD *must* be called (no arguments needed). LASTD sets up the calls to the smoothing routines, KURV1 and KURV2S.

ENTRY POINTS

DASHD, CURVED, FRSTD, VECTD, LINED, PSYMD, LASTD, PWRY, KURV1, KURV2S, CRVED, VECT, FRST

COMMON BLOCKS

```
DASHD1    1478
DASHD2     1
```

I/O

Plots lines.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Bonnie Gacnik, NCAR, Boulder, Colorado 80302

LANGUAGE

FORTTRAN

HISTORY

Written in October 1973, standardized October, 1973.

ALGORITHM The names of the DASHCHAR routines have been changed and the smoothing routines have the names previously found in DASHCHAR routines. This is done so that the user can access DASHSMTH without changing the driver program which had previously been set up to access the routines in DASHCHAR. Points for each line segment are processed and passed to the routines, KURV1 and KURV2S, which compute splines under tension passing through these points. New points are generated between the given points, resulting in smooth lines.

SPACE REQUIRED About 4600₈

ACCURACY ±.5 CRT units per call. There is no cumulative error.

TIMING About three times as long as DASHCHAR.

PORTABILITY An implementor's writeup is available.

PLOTTING ROUTINES USED DASHLN, CURVE, FRSTPT, MXY, VECTOR, PSYM, OPTION

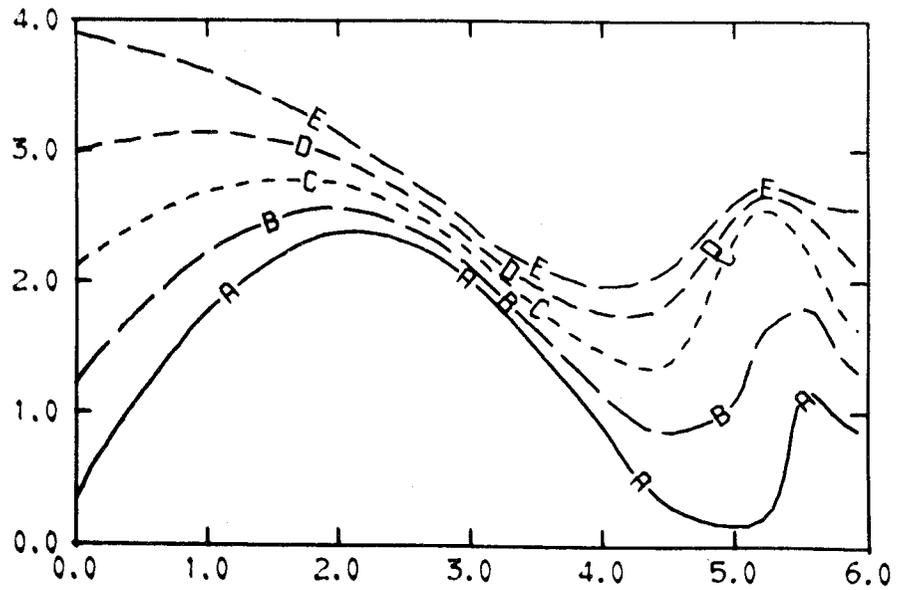
REQUIRED RESIDENT ROUTINES SQRT, ATAN2, SIN, COS, EXP

INTERNAL PARAMETERS

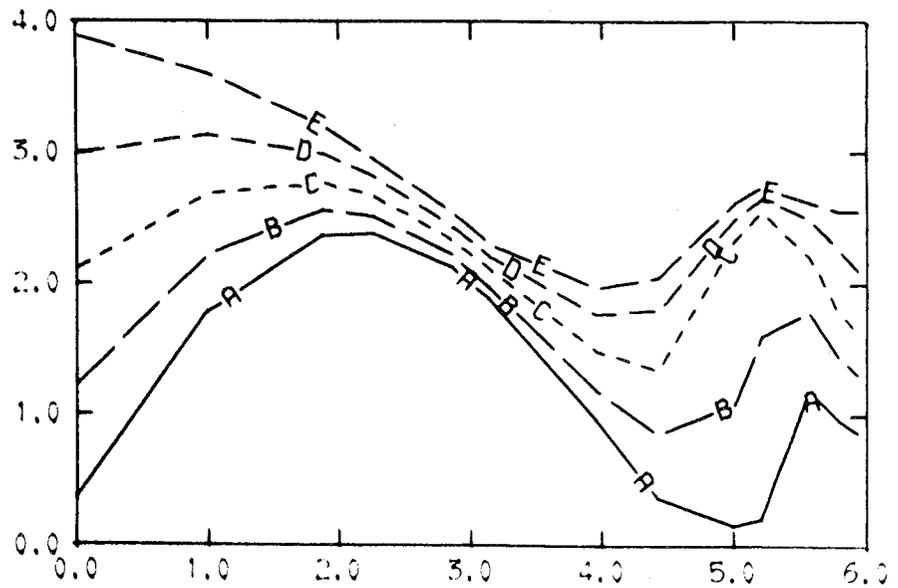
<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
IOFFS	0	Flag to turn on smoothing code. = 0 smoothing. = 1 no smoothing.
TENSN	2.5	Tension factor. Must be greater than 0. A large tension factor (30.) would essentially turn off smoothing.
NP	150	Twice the maximum number of interpolated points on a horizontal line with length equal to that of the grid. More points per unit length are interpolated for short lines than for long lines.
SMALL	4.	The minimum distance in CRT units between points. When points on a line are being processed, and two or more consecutive points are less than 4. CRT units apart, only one of these consecutive points is saved. This procedure is to prevent cusps.
L1	70	The maximum number of points saved at one time. If there are more than 70 points on a given line, 70 points are processed, then the next 70, until the entire line is processed. Smoothness between segments is maintained automatically. If L1 is increased, the dimensions of XSAVE, YSAVE, XP, YP, and TEMP in FRSTD must be increased to the new value of L1.

Below are found figures comparing DASHSMTH with DASHCHAR.

DASHSMTH



DASHCHAR



SUBROUTINE HAFTON (Z,L,M,N,FLO,HI,NLEV,NOPT,NPRM,ISPV,SPVAL)

DIMENSION OF
ARGUMENTS Z(L,M)

LATEST REVISION September 1973

PURPOSE HAFTON draws a half-tone picture from data stored in a rectangular array with the intensity in the picture proportional to the data value.

ACCESS CARDS *FORTRAN,S=ULIB,N=HAFTON
 *COSY

USAGE If the following assumptions are met, use

CALL EZHFTN (Z,M,N)

Assumptions:

All of the array is to be drawn,
Lowest value in Z will be at lowest intensity on reader/printer output,
Highest value in Z will be at highest intensity,
Values in between will appear linearly spaced,
Maximum possible number of intensities are used,
The picture will have a perimeter drawn,
FRAME will be called after the picture is drawn,
Z is filled with numbers that should be used (no unknown values).

USAGE
(continued)

If these assumptions are not met, use

CALL HAFTON (Z,L,M,N,FLO,HI,NLEV,NOPT,NPRM,ISPV,SPVAL)

ARGUMENTS

**On Input
for EZHFTN**

Z
M by N array to be used to generate a half-tone plot.

M
First dimension of Z.

N
Second dimension of Z.

**On Output
for EZHFTN**

All arguments are unchanged.

**On Input
for HAFTON**

Z
The (origin of the) array to be plotted.

L
The first dimension of Z in the calling program.

M
The number of data values to be plotted in the x-direction (the first subscript direction). When plotting all of an array, L = M. See Appendix 1 of the Graphics Chapter for an explanation of using this argument list to process any part of an array.

N
The number of data values to be plotted in the y-direction (the second subscript direction).

FLO
The value of Z that corresponds to the lowest intensity. (When NOPT.LT.0, FLO corresponds to the highest intensity.)
If FLO=HI=0, MIN(Z) is used for FLO.

HI

The value of Z that corresponds to the highest intensity.
 (When NOPT.LT.0, HI corresponds to the lowest intensity.)
 If HI=FLO=0, MAX(Z) is used for HI.

NLEV

The number of intensity levels desired. 16 maximum. If
 NLEV=0 or 1, 16 levels are used.

NOPT

Flag to control the mapping of Z onto the intensities.

The sign of NOPT controls the directness or inverseness of
 the mapping.

- NOPT positive yields direct mapping. That is, the largest Z's produce the densest dots. On mechanical plotters, big Z's will produce a dark area on the paper. With the film development methods used at NCAR, Big Z's will produce many (white) dots on the film, also resulting in a dark area on reader-printer paper.
- NOPT negative yields inverse mapping. That is, the smallest Z's produce the densest dots resulting in dark areas on the paper.

The absolute value of NOPT determines the mapping of Z
 onto the intensities. For IABS(NOPT)

- = 0 The mapping is linear. That is, for each intensity there is an equal range in Z value.
- = 1 The mapping is linear. That is, for each intensity there is an equal range in Z value.
- = 2 The mapping is exponential. That is, for higher Z's there is a larger difference in intensity for relatively close z's. Details in the bigger Z's are displayed at the expense of the smaller Z's.
- = 3 The mapping is logarithmic, so details of smaller Z's are shown at the expense of larger Z's.
- = 4 Sinusoidal mapping, so middle-sized Z's show details at the expense of extreme Z's.
- = 5 Arcsine mapping, so extreme sized Z's are shown at the expense of middle-sized Z's.

On Input
for HAFTON
(continued)

NPRM

Flag to control the drawing of a perimeter around the half-tone picture.

- NPRM = 0: Perimeter drawn with ticks pointing at data locations. (Side lengths proportional to number of data values.)
- NPRM positive: No perimeter, picture fills frame.
- NPRM negative: Picture within confines of users last set call.

ISPV

Flag to tell if the special value feature is being used. The special value feature is used to mark areas where the data is not known or holes are wanted in the picture.

- ISPV = 0: Special value feature not in use. SPVAL is ignored.
- ISPV non-zero: Special value feature in use. SPVAL contains the special value. Where Z contains the special value, no half-tone is drawn. If ISPV
 - = 0 Special value feature not in use. SPVAL is ignored.
 - = 1 Nothing is drawn in special value area.
 - = 2 Contiguous special value areas are surrounded by a polygonal line.
 - = 3 Special value areas are filled with X's.
 - = 4 Special value areas are filled in with the highest intensity.

SPVAL

The value used in Z to mark unknown areas. This argument is ignored if ISPV = 0.

On Output
for HAFTON

All arguments are unchanged.

NOTE

This routine produces a huge number of plotter instructions per picture, averaging over 100,000 point calls per dd80 frame when M = N. The dd80 must be assigned on-line (*ASSIGN,DD80=ONLINE) when more than one or two dozen frames are produced.

ENTRY POINTS	EZHFTN, HAFTON, ZLSET, GRAY, BOUND
COMMON BLOCKS	HAFT01 3 HAFT02 7
I/O	Plots half-tone picture.
PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Thomas Wright, NCAR, Boulder, Colorado 80302
LANGUAGE	FORTRAN
HISTORY	Rewrite of PHOMAP originally written by M. Perry of High Altitude Observatory, NCAR.
ALGORITHM	Bi-linear interpolation on plotter (resolution-limited) grid of normalized representation of data.
SPACE REQUIRED	2442 ₈
TIMING	About 5 seconds on the CDC 6600 for this dd80 version, more or less regardless of the size of Z.
PORTABILITY	An implementors write-up is available.
PLOTTING ROUTINES USED	FRAME, SET, GETSET, PERIM, PWRT, POINT, and LINE.
REQUIRED RESIDENT ROUTINES	None

INTERNAL PARAMETERS

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
XLT	0.1	Left-hand edge of the plot when NSET = 0. (0.0 = left edge of frame, 1.0 = right edge of frame.)
YBT	0.1	Bottom edge of the plot when NSET = 0. (0.0 = bottom of frame, 1.0 = top of frame.)
SIDE	0.8	Length of longer edge of plot (see also EXT).
EXT	.25	Lengths of the sides of the plot are proportional to M and N (when NSET = 0) except in extreme cases, namely, when $\text{MIN}(M,N)/\text{MAX}(M,N)$ is less than EXT. Then a square plot is produced. When a rectangular plot is produced, the plot is centered on the frame (as long as $\text{SIDE}=2*\text{XLT}=\text{SIDE}+2*\text{YBT}=1.$, as with the defaults).
ALPHA	1.6	A parameter to control the extremeness of the mapping function specified by NOPT. (For $\text{IABS}(\text{NOPT}) = 0$ or 1 , the mapping function is linear and independent of ALPHA.) For the non-linear mapping functions, when ALPHA is changed to a number closer to $1.$, the mapping function becomes more linear. When ALPHA is changed to a larger number, the mapping function becomes more extreme.
MXLEV	16	Maximum number of levels. Limited by plotter.
NCRTG	8	Number of CRT units per gray-scale cell. Limited by plotter.
NCRTF	1024	Number of CRT units per frame.
IL	(below)	An array defining which of the available intensities are used when less than the maximum number of intensities are requested.

<u>NLEV</u>	<u>Intensities Used</u>
2	5,11
3	4, 8,12
4	3, 6,10,13
5	2, 5, 8,11,14
6	1, 4, 7, 9,12,15
7	1, 4, 6, 8,10,12,15
8	1, 3, 5, 7, 9,11,13,15
9	1, 3, 4, 6, 8,10,12,13,15
10	1, 3, 4, 6, 7, 9,10,12,13,15
11	1, 2, 3, 5, 6, 8,10,11,13,14,15
12	1, 2, 3, 5, 6, 7, 9,10,11,13,14,15
13	1, 2, 3, 4, 6, 7, 8, 9,10,12,13,14,15
14	1, 2, 3, 4, 5, 6, 7, 9,10,11,12,13,14,15
15	1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15
16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15

A sample picture follows.

INTERNAL PARAMETERS
(continued)



- 1.176
- 1.205
- .8544
- .6634
- .4924
- .5215
- .1505
- 2.2527E-02
- .1915
- .3625
- .5335
- .7045
- .8755
- 1.046
- 1.217

CALL EZHF'FN(Z,30,35)

SUBROUTINE ISOSRF (T,LU,MU,LV,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)**DIMENSION OF
ARGUMENTS**

T(LU,LV,MW),EYE(3),SLAB(MUVWP2,MUVWP2)

LATEST REVISION

January 1974

PURPOSE

ISOSRF draws an approximation of an iso-valued surface from a three-dimensional array with hidden lines removed.

ACCESS CARDS

*FORTRAN,S=ULIB,N=ISOSRF
*COSY

USAGE

If the following assumptions are met, use
CALL EZISOS (T,MU,MV,MW,EYE,SLAB,TISO)

Assumptions:

All of the T array is to be used,
IFLAG is chosen internally,
FRAME is called by EZISOS.

If the assumptions are not met, use
CALL ISOSRF (T,LU,MU,LV,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)

ARGUMENTS

On Input

T

Three dimensional array of data that is used to determine the iso-valued surface.

LU

First dimension of T is the calling program.

MU

The number of data values of T to be processed in the U direction (the first subscript direction). When processing the entire array, $LU=MU$ (and $LV=MV$). See Appendix 1 of the Graphics Chapter for an explanation of using this argument list to process any part of an array.

LV

Second dimension of T in the calling program.

MV

The number of data values of T to be processed in the V direction (the second subscript direction).

MW

The number of data values of T to be processed in the W direction (the third subscript direction).

EYE

The position of the eye in three-space. T is considered to be in a box with opposite corners (1,1,1) and (MU,MV,MW). The eye is at (EYE(1),EYE(2),EYE(3)), which must be outside the box that T is in. While gaining experience with the routine, a good choice for EYE might be $(5.0*MU, 3.5*MV, 2.0*MW)$.

MUVWP2

The maximum of (MU,MV,MW)+2 ($MUVWP2=MAX0(MU,MV,MW)+2$).

SLAB

A work space used for internal storage. SLAB must be at least MUVWP2*MUVWP2 words long.

TISO

The iso-value used to define the surface; the surface drawn will separate volumes of T that have value greater than TISO from volumes of T that have value less than TISO.

IFLAG

A flag which serves two purposes.

- First, the absolute value of IFLAG determines which types of lines are drawn to approximate the surface. Three types of lines are considered: lines of constant U, lines of constant V, and lines of constant W. The following table lists the types of lines drawn.

IABS(IFLAG)	<u>Lines of Constant</u>		
	U	V	W
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
0, 7 or more	Yes	Yes	Yes

See the example picture for further clarification.

- Second, the sign of IFLAG determines what is inside and what is outside, hence, which lines are visible and what is done at the boundary of T. For IFLAG:

positive	T values greater than TISO are assumed to be inside the solid formed by the drawn surface.
negative	T values less than TISO are assumed to be inside the solid formed by the drawn surface.

If the algorithm draws a cube, reverse the sign of IFLAG.

On Output

T,LU,MU,LV,MV,MW,EYE,MUWVP2,TISO, and IFLAG are unchanged.
SLAB has been written in.

NOTE

- This routines is for lower resolution arrays than ISOSRFHR. 40 by 40 by 30 is a practical maximum.
- Transformations can be achieved by adjusting scaling statement functions in ISOSRF, SET3D, and TR32.
- The hidden-line algorithm is not exact, so visibility errors can occur. The sample picture is typical.

ENTRY POINTS

ISOSRF,SET3D,SET32,TRN32,ZEROSC,STCNTR,DRCNTR,FRST,VECT,LAST,TR32,KURV1,KURV2S,FRSTC,VECTC,IVIS,FILLIN,ISOSRB

COMMON BLOCKS

<u>Name</u>	<u>Length</u>
ISOSR1	4
ISOSR2	1323 ₈
ISOSR3	5
ISOSR4	2
ISOSR5	75 ₈
ISOSR6	1015 ₈
ISOSR7	1

I/O

Plots surface

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST Thomas Wright, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Developed for users of ISOSRFHR with smaller arrays.

ALGORITHM Cuts through the three-dimensional array are contoured with a smoothing contourer which also marks a model of the plotting plane. Interiors of boundaries are filled in and the result is .OR.ed into another model of the plotting plane which is used to test subsequent contour lines for visibility.

SPACE REQUIRED About 11000, not including the system plot package.

TIMING Varies widely with size of T and the volume of the space enclosed by the surface drawn. The sample picture took about 1 second of 7600 time.

PORTABILITY An implementor's write-up is available.

PLOTTING ROUTINES USED FRSTPT,VECTOR,FRAME

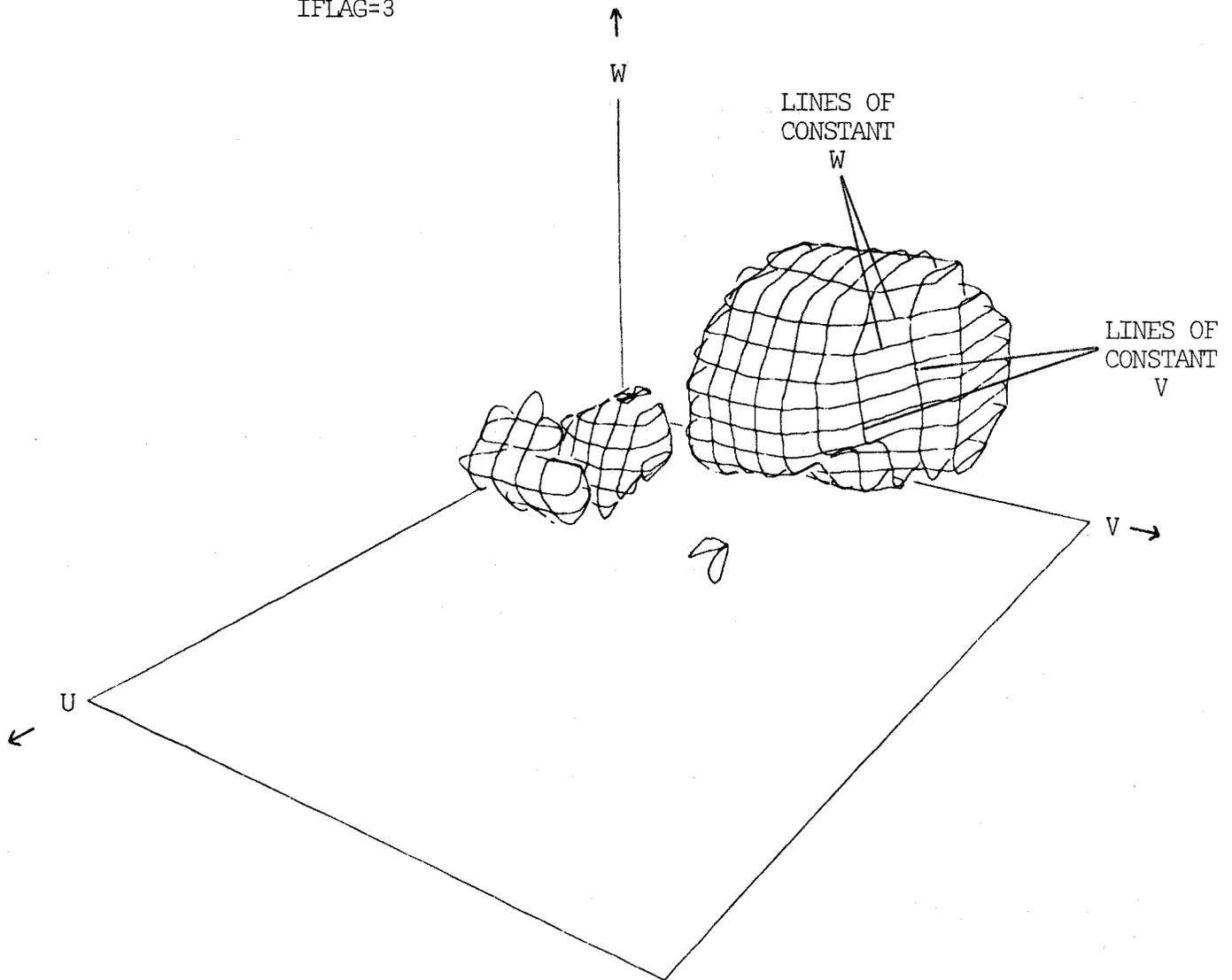
REQUIRED RESIDENT ROUTINES SQRT,ACOS,SIN,COS,ATAN2,EXP,SBYTES,GBYTES

INTERNAL PARAMETERS

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
IREF	1	Flag to control drawing of axes ○ IREF=non-zero means draw axes. ○ IREF=0 means don't.

EXAMPLE

IFLAG=3



ISO-SURFACE PACKAGE FOR HIGH RESOLUTION 3-DIMENSIONAL ARRAYS

DIMENSION OF ARGUMENTS EYE(3),STL(NV,NW,2),IS2(LX,NY),S(4),IOBJS(MV,NW)

LATEST REVISION December 1973

PURPOSE This package of three routines produces a perspective picture of an arbitrary object or group of objects with the hidden parts not drawn. The objects are assumed to be stored in the format described below, a format which was chosen to facilitate the display of functions of three variables (Figure 1) or output from high resolution three-dimensional computer simulations (Figure 2).

ACCESS CARDS *FORTRAN,S=ULIB,N=ISOSRFHR
*COSY

USAGE The object is defined by setting words in a three-dimensional array to one where the object is and zero where it is not. That is, the position in the array corresponds to a position in three-space, and the value of the array tells whether any object is present at that position or not. Because a large array is needed to define objects with good resolution, only a part of the array is passed to the package with each call.

USAGE*(continued)*

There are three subroutines in the package. INIT3D is called at the beginning of a picture. This call can be skipped sometimes if certain criteria are met and certain precautions are taken. See the "Timing" section for details. DANDR (Draw AND Remember) is called successively to process different parts of the three-dimensional array. For example, Figure 3, the nearer plane would be processed in the first call to DANDR, while the further plane would be processed in a subsequent call. An example follows:

```

      CALL INIT3D(EYE,NU,NV,NW,ST1,LX,NY,IS2,IU,S)
      DO 1 IBKWDS=1,NU
        I = NU+1-IBKWDS
C     FORM OR READ SLAB I OF THE 3 DIMENSIONAL ARRAY
C     ONLY 1 OR 0 IN THE SLAB, CALLED IOBJS
      . . .
      1 CALL DANDR(NV,NW,ST1,LX,NX,NY,IS2,IU,S,IOBJS,MV)

```

ARGUMENTS**On Input****EYE**

An array 3 long containing the U, V, and W coordinates of the eye position. Objects are considered to be in a box with 2 extreme corners at (1,1,1) and (NU,NV,NW). The eye position must be positive coordinates away from the coordinate planes U=0, V=0, and W=0. While gaining experience with the package, use EYE(1)=5*NU, EYE(2)=4*NV, EYE(3)=3*NW.

NU

U direction length of the box containing the objects.

NV

V direction length of the box containing the objects.

NW

W direction length of the box containing the objects.

ST1

A scratch array at least NV*NW*2 words long.

LX

The number of words needed to hold NX bits. Also, the first dimension of IS2. See NX and IS2. On CDC 6600 and 7600, $LX=(NX-1)/60+1$.

NX

Number of cells in the x-direction of a model of the image plane. A silhouette of the parts of the picture processed so far is stored in this model. Lines to be drawn are tested for visibility by examining the silhouette. Lines in the silhouette are hidden. Lines out of the silhouette are visible. The solution is approximate because the silhouette is not formed exactly. See IS2.

NY

Number of cells in the y-direction of the model of the image plane. Also the second dimension of IS2.

IS2

An array to hold the image plane model. It is dimensioned LX by NY. The model is NX by NY and packed densely. If hidden lines are drawn, decrease NX and NY (and LX if possible). If visible lines are left out of the picture, increase NX and NY (and LX if need be). As a guide, some examples showing successful choices are listed:

<i>Given</i>	<i>NU</i>	<i>NV</i>	<i>NW</i>	<i>Resulting</i>	<i>NX</i>	<i>NY</i>	<i>From Testing</i>
	100	100	60		200	200	
	60	60	60		110	110	
	40	40	40		75	75	

IU

Unit number of scratch file for the package. ST1 will be written NU times on this file.

S

A real array 4 long which contains the CRT coordinates of the area where the picture is to be drawn. That is, all plotting coordinates generated will be bounded as follows: X coordinates will be between S(1) and S(2), Y coordinates will be between S(3) and S(4). To prevent distortion, have $S(2)-S(1)=S(4)-S(3)$.

$$10.0 \leq S(I) \leq 1010.0 \quad I=1,2,3,4.$$

On Input
(continued)

I OBJ S

A NV by NW array (with actual first dimension MV in the calling program) describing the object. If this is call number 1 to **DANDR**, the part of the picture at $U=NU+1-I$ is to be processed. **I OBJ S** defines the objects to be drawn in the following manner: $I OBJ S(J,K)=1$ if any object contains the point $(NU+1-I,J,K)$ and $I OBJ S(J,K)=0$ otherwise.

MV

Actual first dimension of **I OBJ S** in the calling program. When plotting all of **I OBJ S**, $NV=MV$. See Appendix 1 of the graphics chapter for an explanation of using this argument list to process any part of an array.

On Output

EYE, **NU**, **NV**, **NW**, **LX**, **NX**, **NY**, **IU**, **S**, **I OBJ S**, and **MV** are unchanged. **ST1** and **IS2** have been written in.

NOTE

This routine is for large arrays, $40 \times 40 \times 30$ is a practical minimum.

ENTRY POINTS

INIT3D, **SETORG**, **PERSPC**, and **DANDR**

COMMON BLOCKS

None

I/O

Plots visible surfaces, uses scratch file or tape.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Thomas Wright, NCAR, Boulder, Colorado 80302.

LANGUAGE

FORTRAN

HISTORY

Originally developed at NCAR starting in late 1970.

ALGORITHM The basic method is to contour cuts through the array starting with a cut nearest the observer. The algorithm leaves out the hidden parts of the contours by suppressing lines enclosed within lines produced while processing preceding cuts. The technique is described in detail in the reference cited below.

REFERENCE Wright, T.: A one-pass hidden-line remover for computer drawn three-space objects. *Proc 1972 Summer Computer Simulation Conference*, 261-267, 1972.

SPACE REQUIRED 1474₈

ACCURACY The algorithm is not exact. However, reasonable pictures are produced.

TIMING This routine is very time consuming. If many pictures are produced with the same size arrays and eye position, much time can be saved by rewinding unit IU, filling IS2 with zeros, and skipping the call to INIT3D for other than the first picture.

PORTABILITY Two machine dependent constants are initialized in DANDR. SETORG has an ENTRY statement for PERSPC. In DANDR, .AND. and .OR. are used for masking operations.

PLOTTING ROUTINES USED LINE

REQUIRED RESIDENT ROUTINES SQRT, ACOS, SIN

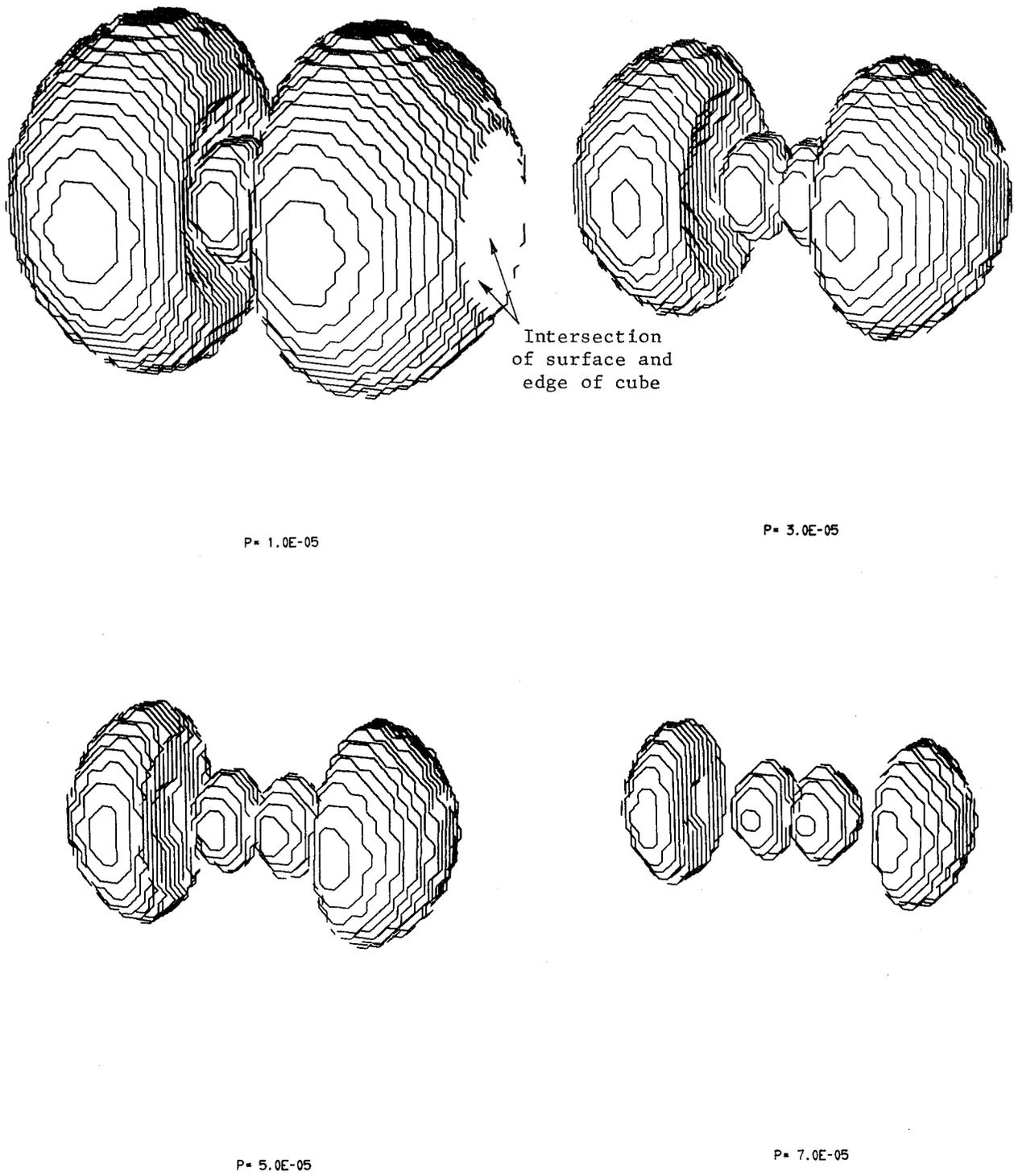


Figure 1. Four contour surfaces of the wave function of a 3-P electron in a one electron atom. $50 \times 50 \times 50$ object cube, 100×100 screen model.

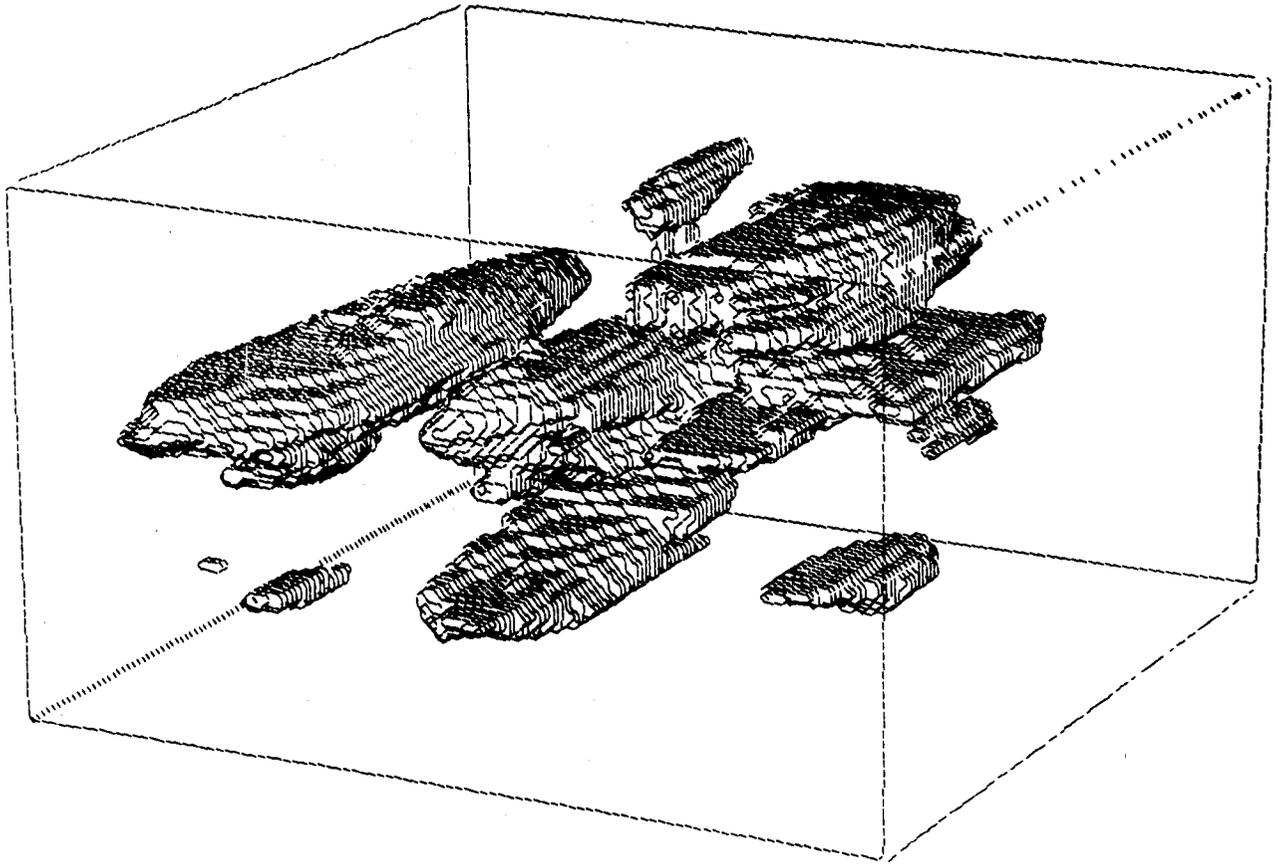


Figure 2. Output from a 3-dimensional cloud model. $100 \times 100 \times 60$ object cube, 200×200 screen model.

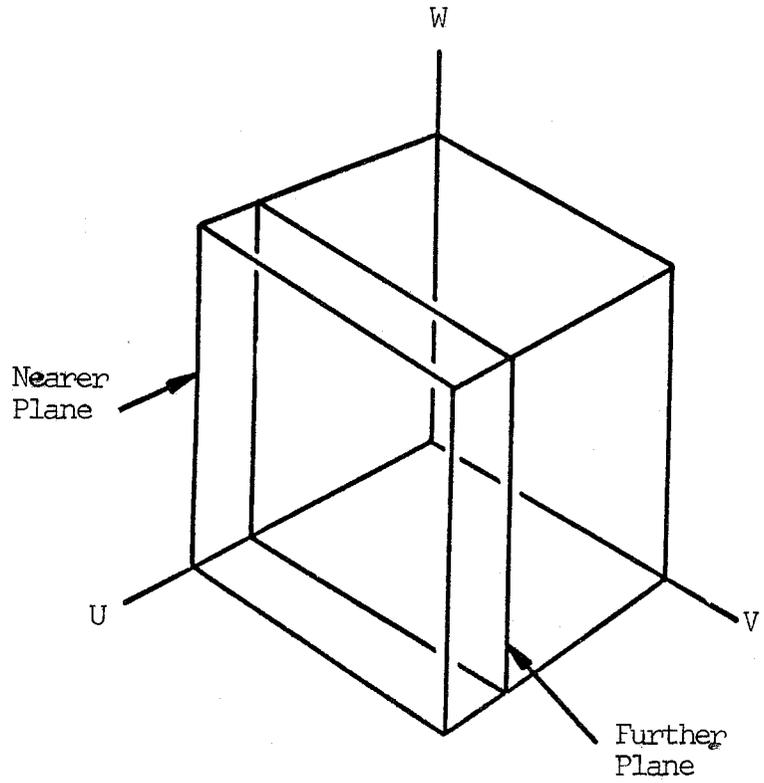


Figure 3. Nearer plane would be processed in first call to DANDR, while the further plane would be processed in a subsequent call.

ARGUMENTS

On Input

X,Y

Positioning coordinates for the characters to be drawn. These can be integers in the range 1 through 1024 or floating point numbers, in which case they are scaled according to the most recent SET call. Also see JCNT.

IDPC

Characters to be drawn and "function codes". (See below.)

N

The number of characters in IDPC, including characters to be drawn and function codes.

SIZ

Size of the character. SIZ may be floating or integer.

- If floating, it is a multiplication factor of digitized character width. (See "function codes" below for these sizes.)
- If SIZ is integer, SIZ is desired CRT units for principal character height (i.e., principal characters will be SIZ CRT units high, and indexical characters will be scaled proportionally, such that

$$\begin{aligned} \text{indexical} &= 13/21 * \text{SIZ CRT units} \\ \text{cartographic} &= 9/21 * \text{SIZ CRT units high.} \end{aligned}$$

IOR

Angle counter clockwise from X axis at which the characters are plotted.

- If integer, $\text{ANGLE} = \text{IOR} * 90$ degrees
- If floating and $0 < \text{IOR} < 2\pi$, IOR is the angle in radians.
- If floating and greater than 6.28, IOR is the angle in degrees.

JCNT

Centering option.

- = 0 (X,Y) is the center of the first character.
- = 1 (X,Y) is the center of the entire string.
- = 2 (X,Y) is the center of the left edge of the first character.
- = 3 (X,Y) is the center of the last character.
- = 4 (X,Y) is the center of the right edge of the last character.

On Output

All arguments are unchanged.

Function Codes

Function codes may be included in the character string IDPC to change font, case, etc. within a string of text. All function instructions must be enclosed in apostrophes. No punctuation is needed between functions except for a comma between adjacent numbers; however, commas may be used between functions to improve readability. The following are the only legal function codes. Any other characters in a function string will be ignored except that an error message will be printed and, if more than 10 errors occur within a string, control will be returned to the main program. At the first call to PWRX, size, type and case are Principal, Roman and upper.

A. Font definitions

- R Roman type characters
- G Greek type characters

B. Size definitions

- P Principal size, digitized to be 21 CRT units high. The total character including white space is 32 CRT units high. A carriage return or a Y increment will space down 32 CRT units. A blank or an X increment will space across 16 CRT units. *Note:* Characters vary in width.

Function Codes
(continued)

I Indexical size, digitized to be 13 CRT units high and 20 CRT units high including white space. A carriage return or a Y increment is 20 CRT units. Blanks or X increments are 12 CRT units.

K Cartographic size, digitized to be 9 CRT units high and 14 CRT units high including white space. Carriage return or Y increments are 14 CRT units. Blanks or X increments are 8 CRT units.

C. Case definitions

U or Un. Upper case

If U is followed by a number n (not separated by a comma) then n characters will be drawn in upper case, subsequent characters will be in lower case. (The Un option is particularly useful for capitalizing sentences.)

L or LN. Lower case

If L is followed by a number n then n characters will be drawn in lower case and subsequent characters will be in upper case.

D. Level definitions

S or Sn. Superscript level

B or Bn. Subscript level.

N or Nn. Normal level.

When super or subscripting, the character size will change depending on the previous character drawn. Principal base characters will be sub or superscripted with indexical characters, with a 10 CRT unit shift (scaled to SIZ) up or down. Indexical and cartographic base characters will be sub or superscripted with cartographic characters with a 7 CRT unit shift.

The case of the indexing characters will remain the same as that of the base character unless otherwise specified, except that a lower case indexical base will be super or subscripted with upper case cartographic, as the cartographic type has no lower case alphabetic or numeric characters available.

If S, B, or N is followed by a number n, then n characters will be drawn as specified above, after which character size, case, and position will be reset to the base character. The N option returns character case and size to the base but maintains the current character position.

Example: 'Ul'T'Sl'est
 will be written T^est
 'Ul'T'S'e'N'st
 will be written T^est

E. Coordinate definitions (Descriptions assume normal CRT unit space.)

H,HN,H,Q. Increment in the X direction.

If this option appears without a number n, n will be taken to be 1. Hn will shift the present X position n CRT units. If n is positive the shift is to the right, if n is negative the shift is to the left. If Hn is followed by a Q, the X position will be incremented by n character widths (i.e., n blanks) either right or left.

V,Vn,VnQ. Increment in the Y direction.

If this option appears without a number n, n will be taken to be 1. Vn will shift the present Y position n CRT units. If n is positive the shift is up, if n is negative the shift is down. If Vn is followed by a Q, the Y position will be incremented by n lines up or down.

X,Xn. Set X.

If X appears without a number n, this will act as a do nothing statement. Otherwise, the character position in the X direction will be set to the CRT coordinate n, so that the next character drawn will be centered on n and subsequent characters will be drawn from this position.

Y,Yn. Set Y.

This works the same as set X.

C

A carriage return will be done before the next character is plotted.

F. Direction definitions

D,Dn. Write down, rather than across the frame.

If D appears without an n, all characters will be written down, until an 'A' function is encountered. If D is followed by a number n, n characters will be written down and subsequent characters will be written across the frame.

Function Codes
(continued)

A

Write across. Escape from D option.

G. Direct character access.

nnn. Numeric character.

Character number nnn will be drawn. nnn is base 8.

NOTE

- All characters in a given call are drawn in the same intensity. If $SIZ \leq 1.5$, characters are in low intensity, otherwise they are in high intensity. Return to the main program is always in high intensity.
- If the multiplication factor for drawing characters is less than .01 or greater than 100.0, it will be set to 1.0.
- On other than the first entry to PWRX, font, case, etc. are in the state last assumed in the previous call.

ENTRY POINTSPWRX,GINUM,GETNUMB,XTCH,GTCH,DAT1,DAT2,DAT3,DAT4,DAT5,DAT6,
DAT7,DAT8**COMMON BLOCKS**PWRXCM 1455₈**I/O**

Plots characters.

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST Thomas Wright and J. W. Chalmers, NCAR, Boulder, Colorado 80303.

LANGUAGE FORTRAN

HISTORY Implemented by Dori Bundy to make Hershey's character set more useable.

ALGORITHM Digitizations of the characters are stored internally and adjusted according to X, Y, SIZ, IOR, and JCNT, then plotted.

SPACE REQUIRED 7151₈

TIMING It takes a different amount of time to draw different characters. The whole character set can be drawn in approximately 1 sec CPU on the 7600.

PORTABILITY This routine is coded for a 60-bit word length.

PLOTTING ROUTINES REQUIRED FRSTPT, VECTOR, MXY, and OPTION

REQUIRED RESIDENT ROUTINES COS, SQRT

PWRX CHARACTER SET

The following examples show all the characters available to PWRX. Characters may be accessed in two ways. The set may be declared in a function code, then the desired character in that set is determined by KP, a keypunch character. For example, 'KGL'A would produce a @.

A character may be specified numerically. The numeric representation of a character is found by adding the appropriate number 'NUM' to the base number for the set in which the desired character is located. For example, '1301' would produce a @.

SET	PRU	PRL	IRU	IRL	KRU	KRL	PGU	PGL	IGU	IGL	KGU	KGI,	
BASE	0	100	200	300	400	500	600	700	1000	1100	1200	1300	
KP NUM													
A	1	A	a	A	a	^	@	A	α	A	α	^	@
B	2	B	b	B	b	B	§	B	β	B	β	B	§
C	3	C	c	C	c	C	†	Γ	γ	Γ	γ	Γ	†
D	4	D	d	D	d	D	‡	Δ	δ	Δ	δ	Δ	‡
E	5	E	e	E	e	E	⊙	E	ε	E	ε	E	⊙
F	6	F	f	F	f	F	♀	Z	ζ	Z	ζ	Z	♀
G	7	G	g	G	g	G	⊕	H	η	H	η	H	⊕
H	10	H	h	H	h	H	♂	Θ	ϑ	Θ	ϑ	Θ	♂
I	11	I	i	I	i	I	—	I	ι	I	ι	I	ι
J	12	J	j	J	j	J	/	K	κ	K	κ	K	κ
K	13	K	k	K	k	K		Λ	λ	Λ	λ	Λ	λ
L	14	L	l	L	l	L	\	M	μ	M	μ	M	μ
M	15	M	m	M	m	M	—	N	ν	N	ν	N	ν

SET=	PRU	PRL	IRU	IRL	KRU	KRL	PGU	PGL	IGU	ICL	KGU	KGL
BASE=	0	100	200	300	400	500	600	700	1000	1100	1200	1300
KP NUM												
N 16	N	n	N	n	N	/	Ξ	ξ	Ξ	ξ	:	'
O 17	O	o	O	o	O	/	Ο	ο	Ο	ο	ο	'
P 20	P	p	P	p	P		Π	π	Π	π	π	'
Q 21	Q	q	Q	q	Q	\	Ρ	ρ	Ρ	ρ	Ρ	'
R 22	R	r	R	r	R	\	Σ	σ	Σ	σ	Σ	'
S 23	S	s	S	s	S	-	Τ	τ	Τ	τ	Τ	'
T 24	T	t	T	t	T	/	Υ	υ	Υ	υ	Υ	'
U 25	U	u	U	u	U		Φ	φ	Φ	φ	Φ	'
V 26	V	v	V	v	V	\	Χ	χ	Χ	χ	Χ	'
W 27	W	w	W	w	W	ο	Ψ	ψ	Ψ	ψ	Ψ	'
X 30	X	x	X	x	X	□	Ω	ω	Ω	ω	Ω	'
Y 31	Y	y	Y	y	Y	Δ	○	○	○	○	○	'
Z 32	Z	z	Z	z	Z	◇	◦	◦	§	~	ℜ	ϑ

SET=	PRU	PRL	IRU	IRL	KRU	KRL	PGU	PGL	IGU	ICL	KGU	KGL	
BASE=	0	100	200	300	400	500	600	700	1000	1100	1200	1300	
KP NUM													
0	33	0	:	0	:	0	:	0	→	0	→	0	0
1	34	1	;	1	;	1	;	1	↑	1	↑	1	1
2	35	2	!	2	!	2	!	2	←	2	←	2	2
3	36	3	?	3	?	3	?	3	↓	3	↓	3	3
4	37	4	×	4	×	4	×	4	,	4	,	4	4
5	40	5	.	5	.	5	.	5	'	5	'	5	5
6	41	6	÷	6	÷	6	▣	6	'	6	'	6	6
7	42	7	≡	7	≡	7		7	,	7	,	7	7
8	43	8	<	8	<	8	⊥	8	'	8	'	8	8
9	44	9	>	9	>	9	∠	9	"	9	"	9	9
+	45	+	≅	+	≅	+	∴	&	#	&	#	•	•
-	46	-	≅	-	≅	-	☆	∩	∃	∩	∃	■	┌
*	47	*	∅	*	∅	*	▽	∪		∪		▲	∧

SET= PRU PRL IRU IRL KRU KRL PGU PGL IGU IGL KCU KCL
 BASE= 0 100 200 300 400 500 600 700 1000 1100 1200 1300

KP NUM

/	50	/	▽	/	▽	/	▽	∩		∩		◀	
(51	([([({	⌋	⌈	⌋	⌈	<	{
)	52)])])	}	⌌	⌋	⌌	⌋	>	}
\$	53	\$	✓	\$	✓	\$	√	∩	≠	∩	≠	▶	✓
=	54	=	∫	=	∫	=	∫	€	⊕	€	⊕	▶	∫
	55												
,	56	,	∞	,	∞	,	∞	∞	±	∞	±	*	≡
.	57	.	%	.	%	.	°	~	≠	~	≠	▶	---

SUBROUTINE PWRY (X,Y,ID,N,ISIZE,ITHETA,ICNT)

DIMENSION OF ARGUMENTS ID((N-1)/10+1) (10 = number of characters per word)

LATEST REVISION November 1972

PURPOSE PWRY is a character plotting routine. It has some features not found in PWRT, but is not as fancy as PWRX.

ACCESS CARDS *FORTRAN,S=ULIB,N=PWRY
*COSY

USAGE CALL PWRY (X,Y,ID,N,ISIZE,ITHETA,ICNT)

ARGUMENTS

On Input X,Y
Positioning coordinates for the characters to be drawn. These can be integers in the range 1 through 1024 or floating point numbers, in which case they are scaled according to the most recent SET call. Also see ICNT.

ID
Characters to be drawn.

N
The number of characters in ID.

On Input
(continued)**ISIZE**

Size of the character. ISIZE may be floating or integer.

- If floating, it is a multiplication factor of a 6 CRT character width.
- If integer between 0 and 3, the factor is chosen as in PWRT (1., 1.5, 2., 3. times the 6 CRT width).
- If integer greater than 3, it is the character width in CRT units.

ITHETA

Angle counter clockwise from X axis that the characters are plotted at.

- If integer, $ANGLE = ITHETS * 90$ degrees.
- If floating, $ANGLE = ITHETS$ radians.

ICNT

Centering option.

0 (X,Y) is the center of the first character.

1 (X,Y) is the center of the entire string.

2 (X,Y) is the center of the left edge of the first character.

3 (X,Y) is the center of the last character.

4 (X,Y) is the center of the right edge of the last character.

On Output

All arguments are unchanged.

ENTRY POINTS

PWRY

COMMON BLOCKS

None

I/O

Plots characters.

PRECISION	Single
REQUIRED ULIB ROUTINES	None
SPECIALIST	Thomas Wright, NCAR, Boulder, Colorado 80303
LANGUAGE	FORTRAN
HISTORY	Implemented for use in DASHCHAR.
ALGORITHM	Digitizations of the characters are stored internally and adjusted according to X, Y, ISIZE and ICNT, then plotted.
SPACE REQUIRED	510 ₈
TIMING	Slower than PWRT, faster than PWRX.
PORTABILITY	This routine is coded for a 60-bit word length. A portable version is also available.
PLOTTING ROUTINES USED	FRSTPT, VECTOR, and MXY
REQUIRED RESIDENT ROUTINES	COS, SIN

Example

A sample output follows:

CALL PWRY(MX,MY, ID,N,3,0,0)
CALL PWRY(MX,MY, ID,N,2,0,0)
CALL PWRY(MX,MY, ID,N,1,0,0)
CALL PWRY(MX,MY, ID,N,0,0,0)

SIZE=6.5

1-ALPHA I THETA=1
0-ALPHA I THETA=0
2-ALPHA I THETA=2
3-ALPHA I THETA=3

THETA=0.87
CENTRED
CENTRED

SUBROUTINE PWRZ (X,Y,Z, ID,N, ISIZE,LINE,ITOP,ICNT)**DIMENSION OF
ARGUMENTS**

ID((N-1)/10+1) (10 = number of characters per word)

LATEST REVISION

November 1972

PURPOSE

PWRZ is a character plotting routine for plotting characters in three-space when using ISOSRF or SRFACE. (See picture.) For a large class of possible positions, the hidden character problem is solved.

ACCESS CARDS

*FORTRAN,S=ULIB,N=PWRZ
*COSY

USAGE

CALL PWRZ (X,Y,Z, ID,N, ISIZE,LINE,ITOP,ICNT)

Use CALL PWRZ after calling ISOSRF or SRFACE and before calling FRAME.

Note: The SRFACE will have to be changed to suppress the FRAME call. See IFR in SRFACE Internal Parameters.

ARGUMENTS

On Input

X,Y,Z

Positioning coordinates for the characters to be drawn. These are floating point numbers in the same three-space as used in ISOSRF or SRFACE. (Called U, V, W in ISOSRF.)

ID

Characters to be drawn.

N

The number of characters in ID.

ISIZE

Size of the character. ISIZE may be floating or integer. The characters assume the following sizes if viewed "head on". If viewed from an angle, they look smaller.

- If floating, it is a multiplication factor of a 6 CRT character width.
- If integer between 0 and 3, the factor is chosen as in PWRT (1., 1.5, 2., 3. times the 6 CRT width).
- If integer greater than 3, it is the character width in CRT units.

LINE

The direction in which the characters are to be written.

1 = +X	-1 = -X
2 = +Y	-2 = -Y
3 = +Z	-3 = -Z

ITOP

The direction from the center to the top of the first character. Note that LINE cannot equal ITOP even in absolute value. (In the sample picture, "X-Y PLANE" had LINE=2, ITOP=-1.)

ICNT

Centering option.

- 0 (X,Y,Z) is the center of the first character.
- 1 (X,Y,Z) is the center of the entire string.
- 2 (X,Y,Z) is the center of the left edge of the first character.
- 3 (X,Y,Z) is the center of the last character.
- 4 (X,Y,Z) is the center of the right edge of the last character.

On Output

All arguments are unchanged.

NOTE

The hidden character problem is solved correctly for characters near (but not inside) the three-space object.

ENTRY POINTS

PWRZ, INITZ, VISSET, FRSTZ, VECTZ

COMMON BLOCKS

PWRZ1 7

I/O

Plots character.

PRECISION

Single

REQUIRED ULIB
ROUTINES

ISOSRF or SRFACE

SPECIALIST

Thomas Wright, NCAR, Boulder, Colorado 80303

12.PWRZ.4

LANGUAGE

FORTRAN

HISTORY

Implemented for use with ISOSRF and SRFACE.

ALGORITHM

Digitizations of the characters are stored internally and adjusted according to X, Y, Z, ISIZE, LINE, ITOP and ICNT, then plotted.

SPACE REQUIRED

About $1000_8 = 512_{10}$

TIMING

Slightly slower than PWRY.

PORTABILITY

This routine is coded for a 60-bit word length.
A portable version is also available.

PLOTTING ROUTINES
USED

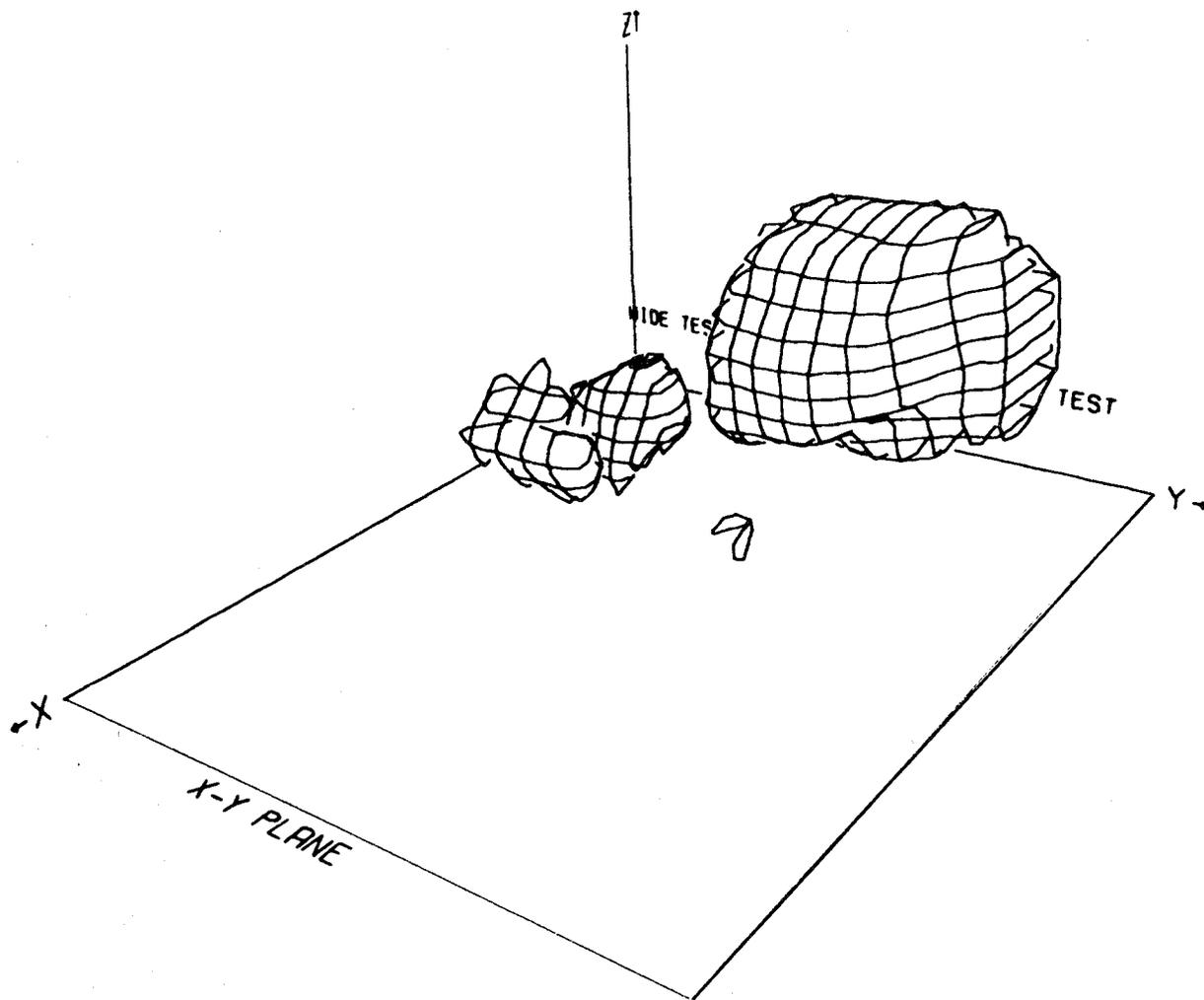
FRSTPT, VECTOR

REQUIRED RESIDENT
ROUTINES

COS, SIN

Example

A sample output follows.



SCROLL**SUBROUTINE SCROLL (IBF,LEN,CARDS,NCARDS,NYST,NYFIN,TST,TMV,TFIN,MOVIE)**

DIMENSION OF ARGUMENTS IBF(LEN),CARDS(8,NCARDS)

LATEST REVISION November 1973

PURPOSE To produce scrolled movie titles with a minimum of effort.

ACCESS CARDS *FORTRAN,S=ULIB,N=SCROLL
 *COSY
 *FORTRAN,S=ULIB,N=PWRX
 *COSY

USAGE If the following assumptions are met, use

CALL MTITLE (PBUF,ISIZE,MOVIE)

Assumptions:

Each group of lines of text is contained on one frame (i.e., no scrolling).

Vertical spacing of titles is done automatically.

Titles are centered horizontally.

There is a maximum of 60 characters per line of text including PWRX modifiers.

In production mode, blank frames are generated before and after each group of title frames.

If these assumptions are not met, use

CALL SCROLL (IBF,LEN,CARDS,NCARDS,NYST,NYFIN,TST,TMV,TFIN,
MOVIE)

ARGUMENTS

For MTITLE the argument list and data input are different from SCROLL and are described separately below.

**On Input
For MTITLE**Parameters

PBUF

Scratch area dimensioned ISIZE used as buffer for plotting instructions. Typically 5000₁₀ words are required.

ISIZE

Dimension of PBUF.

MOVIE

Switch to indicate whether this is a practice run or the movie is being made.

= 0 Movie is being made.

≠ 0 Practice run.

Practice runs output an outlined frame of titles with a legend indicating how many seconds the frame will be shown. The number of blank frames that will be output before and after the title sequence are also indicated. When the movie is being made this practice output is suppressed.

Data Cards

Each group of data cards results in a frame of titles which is repeated to give enough time for reading. There can be any number of groups. MTITLE keeps processing groups until an NCARD=0 or EOF is read. A group consists of the following:

- A header card from which NCARD, TIME, SIZE are read under FORMAT (I5,2F5.1).

NCARD

Number of text cards that follow. If NCARD=0 MTITLE will return to the calling routine.

TIME

Time in seconds this frame should be displayed.

SIZE

Relative size of characters. This multiplies the PWRX character height.

- Text cards containing the characters to be displayed as movie titles and PWRX modifiers (see PWRX writeup). The characters should be left justified and not exceed column 60.

**On Output
For MTITLE**

Workspace plot buffer has been consumed. Other arguments are unchanged.

**On Input
For SCROLL**

IBF

Scratch area dimensioned LEN used as buffer for plotting instructions. Typically 5000₁₀ words are required.

LEN

Dimension of IBF.

CARDS

An 8 by NCARDS array which the user has filled prior to calling SCROLL (either by internal manipulations or by reading cards). CARDS(8,NCARDS) is to contain the following data in card images:

Columns 1-5

The MX coordinate of this line of text on the scroll, or an indicator that this line of text is a continuation of the previous line. MX is the coordinate of the middle of the line if ICNTR is 1, and it is the coordinate of the center of the first character if ICNTR is 0. See columns 11-15 for ICNTR.

The scroll of text is 1024 CRT units wide and any number of CRT units high. MX=-9999 is the continue card indicator. Allowance is made to continue a line of text on up to 3 continue cards.

Columns 6-10

The MY coordinate of this line of text on the scroll (MY may be outside the 1-1024 CRT unit range). In the case of a continue card columns 6-20 are ignored.

**On Input
For SCROLL**
(continued)

Columns 11-15

ICNTR the centering option:

= 0 Start the text at MX.

= 1 Center the text about MX.

Columns 16-20

SIZE, the relative size of characters. This multiplies the PWRX character height. Recommended range 1. - 2.5 (PWRX modifiers can also be used to change sizes).

Columns 21-80

Text for this line, or for continuation of a line when MX=-9999. Note: every line of text, except continuation lines, must start with a PWRX modifier if any PWRX modifiers are used in any text. The status of PWRX will not necessarily be in the mode of the preceding line of text.

NCARDS

Second dimension of CARDS, i.e., the number of card images in CARDS.

NYST

The scroll coordinate that will be at the center of the screen when the text is first displayed (see diagram for clarification).

NYFIN

As NYST but for final position.

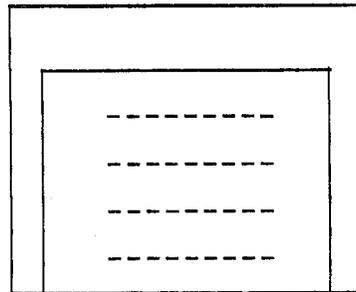
TST

Time in seconds that the scroll will be stationary at NYST. One second is recommended.

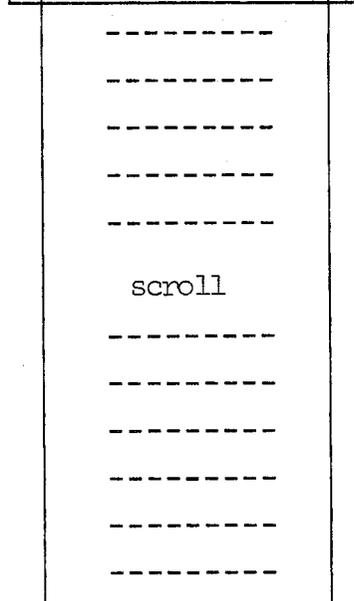
TMV

Time to move the scroll from NYST to NYFIN. This should be the time required to read the text aloud at slow to normal speed.

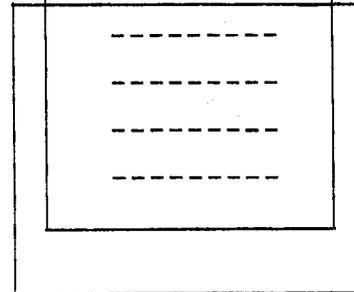
view in screen initially



← scroll coordinate
NYST at center of
screen for TST secs.



scroll moves from
NYST to NYFIN in
TMV secs.



← scroll coordinate
NYFIN at center of
screen for TFIN secs.

final view in screen

**On Input
For SCROLL
(continued)**

TFIN

Time that the scroll will be stationary at NYFIN. One second is recommended.

MOVIE

Switch to indicate whether this is a practice run or the movie is being made.

= 0 Movie being made.

≠ 0 Practice run.

Practice runs output outlined representative frames from the scroll with a legend indicating the number of seconds the frame will be shown at the start or finish, or the number of seconds into the total moving time that a particular frame represents. When the movie is being made this practice output is suppressed.

**On Output
For SCROLL**

Workspace plot buffer has been consumed. Other arguments are unchanged.

NOTE

- See the internal parameter list below for directions for the most common modifications.
- SCROLL contains a general purpose windowing package.

ENTRY POINTS

MTITLE, GAP, SCROLL, WNDOUT, DOTEDGE, PWRW, FTOI, FRSTW, VECTW, PWRX, GTNUM, GTNUMB, XTCH, GTCH, DAT1, DAT2, DAT3, DAT4, DAT5, DAT6, DAT7, DAT8

COMMON BLOCKS

In SCROLL	AUTOT	1
	WINDOW	4
In PWRW	PWRXCM	5125B

I/O

MTITLE

Reads data from cards and outputs dd80 frames.

SCROLL

Is passed all its input through the argument list and outputs dd80 frames.

PRECISION	Single
REQUIRED ULIB ROUTINES	PWRX
SPECIALIST	Pat Coyle, NCAR, Boulder, Colorado 80302
LANGUAGE	FORTRAN
HISTORY	Improves and refines old routines AUTOTITLE and SCROLL at NCAR.
ALGORITHM	<p>SCROLL in effect moves the body of text up through the screen window outputting the frames required to generate a movie sequence of duration specified by the user.</p> <p>At each frame SCROLL skips plotting lines of text that are completely outside of the screen window, plots lines of text entirely within the window using PWRX, and for text lines that are partially within the window uses PWRW. PWRW is similiar to PWRX except that it allows the portions of characters that are only partially within the window to be drawn without distortion. Note: PWRW uses a generalized windowing capability provided by SUBROUTINE FRSTW, which may be used separately in other plotting applications where windowing is desirable.</p>
SPACE REQUIRED	About 14000, not including the user's plot buffer or system plot package.
TIMING	Varies widely.
PORTABILITY	Not presently.

PLOTTING ROUTINES USED FRAME, FLASH1, FLASH2, FLASH3, LINE, FRSTPT, VECTOR, PWRT, OPTION, GETSET

REQUIRED RESIDENT ROUTINES SIN, COS, SQRT, LOG

INTERNAL PARAMETERS

In	MTITLE	<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
		PCHSZ	32.	Height of PWRX character size for a given font (currently PRU).
		GAPSZ	40.	Value of interline spacing for the above font.
		T1	1.	Number of seconds worth of blank frames generated before any title frames are produced (at 24 frames/second).
		T2	.5	Number of seconds worth of blank frames generated between sets of title frames and after the last set of title frames.
In	SCROLL	ICC(n)	n=24	ICC is continue buffer of length n words or 10n characters. This currently allows for a text card plus 3 continue cards (columns 21-80/card).
		NXST	512	Analogous to NYST and NYFIN in argument list. Allows for limited scrolling in the X or horizontal direction. NXST and NXFIN are confined to be within X limits of the window, with the additional constraint that text must leave the window through the top rather than the sides.
		NXFIN	512	
		ICRTJP	300	MY scroll coordinate spacing between practice output frames.
		LIM(4)	(1,1024, 1,1024)	Contains the 4 values that define the window. Set in FRSTW and linked via COMMON/WINDOW/.

Example

This code illustrates how MTITLE is used.

```

      .
      .
      .
      DIMENSION PBUF(5000)
      MOVIE=1
C     FOR PRACTICE RUN
      .
      .
      CALL MTITLE (PBUF,3000,MOVIE)
      .
      .

```

Data Cards

```

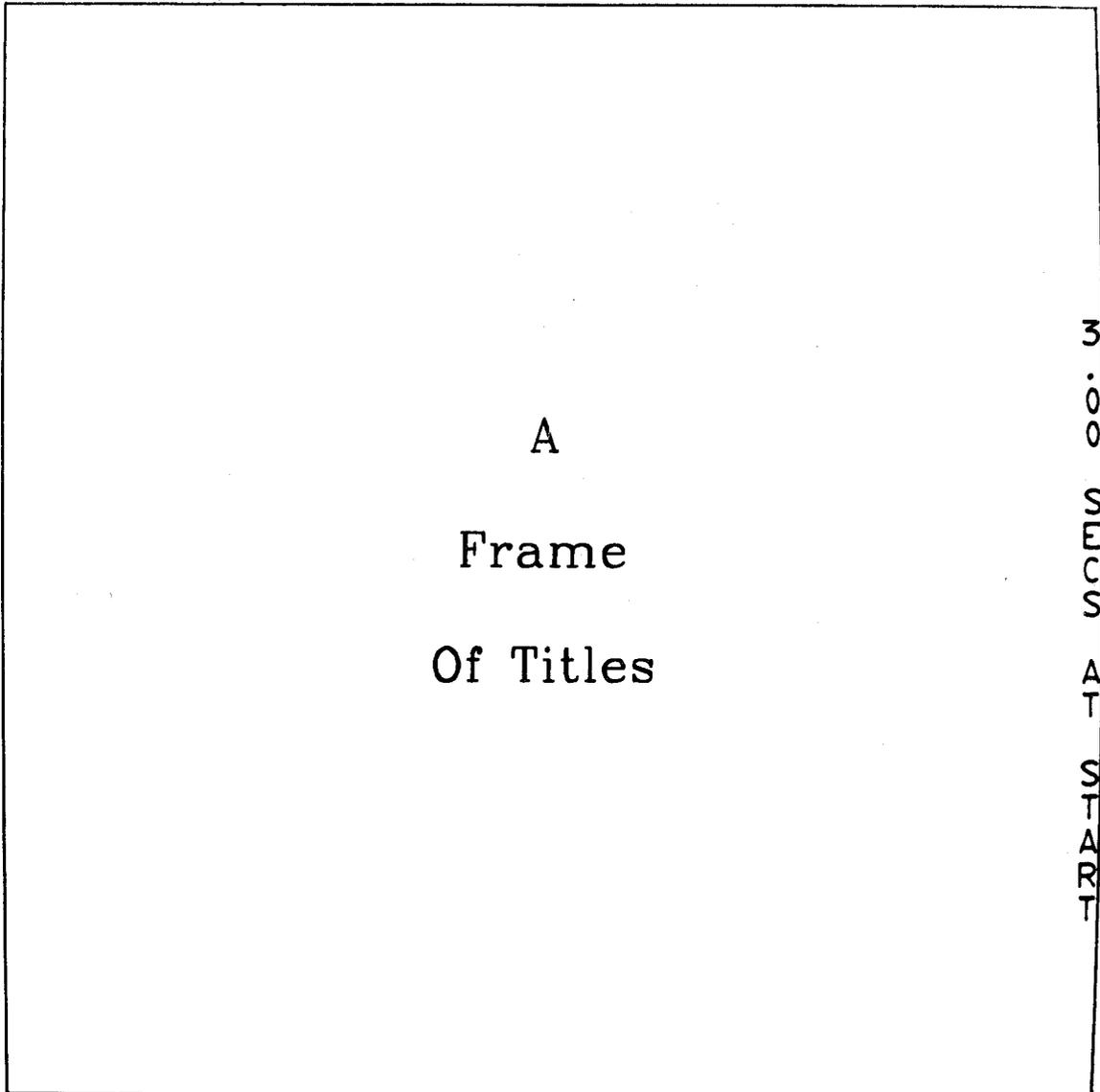
      3  3.  1.5
'PRU'A
'PRU'F'L'RAME
'PRU'O'L'F 'PRU'T'L'ITLES

      6  5.  1.25
'PRU'P'L'RODUCED BY
'PRU'T'L'HE 'U1'NATIONAL 'U1'CENTER FOR
'PRU'ATMOSPHERIC 'U1'RESEARCH'U1',
'PRU'BOULDER'U1', 'U1'COLORADO 'U'80302
'PRU'U'L'NDER SPONSORSHIP OF THE
'PRU'N'L'ATIONAL 'PRU'S'L'CIENCE 'PRU'F'L'OUNDATION
      .

```

Example
(continued)

The output produced is shown below.



Example
(continued)

Produced by
The National Center for
Atmospheric Research,
Boulder, Colorado 80302
Under sponsorship of the
National Science Foundation

5
0
0
S
E
C
S
A
T
S
T
A
R
T

SUBROUTINE SRFACE (X,Y,Z,M,MX,NX,NY,S,STEREO)**DIMENSION OF
ARGUMENTS**

X(NX),Y(NY),Z(MX,NY),M(2,NX,NY),S(6)

LATEST REVISION

January 1973

PURPOSE

SRFACE draws a perspective picture of a function of two variables with hidden lines removed. The function is approximated by a two-dimensional array of heights.

ACCESS CARDS

*FORTRAN,S=ULIB,N=SRFACE
*COSY

USAGE

If the following assumptions are met, use

CALL EZSRFC (Z,M,N,ANGH,ANGV,WORK)

Assumptions:

The entire array is to be drawn,
The data is equally spaced (in the x-y plane),
No STEREO pairs,
Scaling is chosen internally.

If these assumptions are not met use

CALL SRFACE (X,Y,Z,M,MX,NX,NY,S,STEREO)

ARGUMENTS

On Input
for EZSRFC

Z
The M by N array to be drawn.

M
The first dimension of Z.

N
The second dimension of Z.

ANGH
Angle in degrees in the x-y plane to the line of sight
(counter-clock wise from the plus-x axis).

ANGV
Angle in degrees from the x-y plane to the line of sight
(positive angles are above the middle Z, negative below).

WORK
A scratch storage dimensioned at least $2 \times M \times N + M + N$.

On Output
for EZSRFC

Z, M, N, ANGH, ANGV are unchanged. WORK has been written
in.

On Input
for SRFACE

X
A linear array NX long containing the X coordinates of
the points in the surface approximation. See NOTE, page 4.

Y
The linear array NY long containing the Y coordinates of
the points in the surface approximation. See NOTE, page 4.

Z
An array MX by NY containing the surface to be drawn in
NX by NY cells. $Z(I,J) = F(X(I),Y(J))$. See NOTE, page 4.

M

Scratch array at least $2 \times NX \times NY$ words long.

MX

First dimension of Z.

NX

Number of points in the X direction in Z. When plotting an entire array, $MX=NX$. See Appendix 1 of the Graphics Chapter for an explanation of using this argument list to process any part of an array.

NY

Number of points in the Y direction in Z.

S

S defines the line of sight. The viewers eye is at (S(1), S(2), S(3)) and the point looked at is at (S(4), S(5), S(6)). The eye should be outside the block with opposite corners (X(1), Y(1), ZMIN) and (X(NX), Y(NY), ZMAX) and the point looked at should be inside it. For a nice perspective effect, the distance between the eye and the point looked at should be 5 to 10 times the size of the block. See NOTE, page 4.

STEREO

Flag to indicate if STEREO pairs are to be drawn. 0 means no STEREO pair (one picture). Non-zero means put out two pictures. The value of STEREO is the relative angle between the eyes. A value of 1.0 produces standard separation. Negative STEREO reverses the left and right figures.

On Output
for SRFACE

X, Y, Z, MX, NX, NY, S, STEREO are unchanged. M has been written in.

NOTE

- The range of Z compared with the range of X and Y determines the shape of the picture. They are assumed to be in the same units and not wildly different in magnitude. S is assumed to be in the same units as X, Y, and Z.
- Picture size can be made relative to distance. See SETR comments.
- TRN32P can be used to translate from 3 space to 2 space. See comments in SET32P.
- Data with extreme discontinuities may cause visibility errors. If this problem occurs, use a distant eye position away from the +Z axis.

ENTRY POINTS

SRFACE, DRAW, MARK, DRMK, SET32F, TRN32F, SET32P, TRN32P, CLSET, CTCCELL, SETR, SRFACD, EZSRFC

COMMON BLOCKS

SRFBLK (4066 octal in length)

I/O

Plots

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

Thomas Wright, NCAR, Boulder, Colorado 80302

LANGUAGE

FORTRAN

HISTORY

Replaces K.S.+G. algorithm called solids at NCAR. Written December 1971, standardized January 1973.

ALGORITHM Highest so far is visible from above. (See reference.)

REFERENCE Wright, T.J., A Two Space Solution to the Hidden Line Problem for Plotting a Function of Two Variables. IEEE Trans. Comp., pp 28-33, January 1973.

SPACE REQUIRED 11607 octal including SRFBLK, not including plot package.

ACCURACY If the ends of a line segment are visible, the middle is assumed visible.

TIMING Proportional to $NX*NY$. 20 by 20 takes 0.5 seconds on CDC 6600.

PORTABILITY A one page implementors writeup is available.

PLOTTING ROUTINES USED LINE, FRAME

REQUIRED RESIDENT ROUTINES SIN, COS, SQRT, ACOS, ALOG10

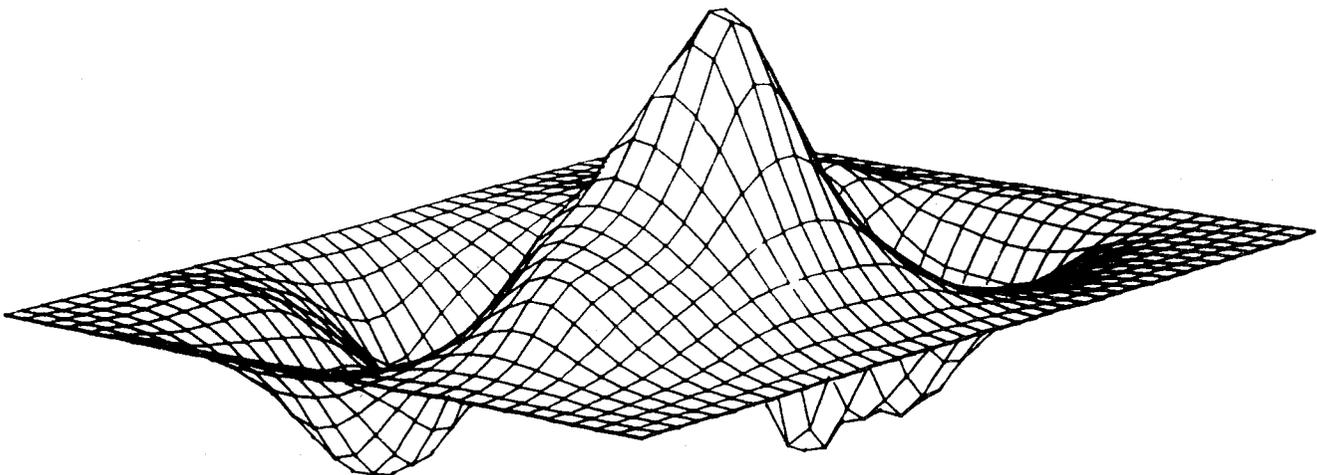
INTERNAL PARAMETERS	<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
	IFR	1	-1 Call FRAME first. 0 Do not call FRAME. +1 Call FRAME when done.
	ISTP	0	STEREO type if STEREO non-zero. -1 Alternating frames, slightly offset (for movies. IROTS=0). 0 Blank frame between (for STEREO slide. IROTS=1). +1 Both on same frame. (Left picture to left side. IROTS=0).
	IROTS	0	0 +Z in vertical plotting direction (CINE mode). +1 +Z in horizontal plotting direction (COMIC mode).
	IDRX	1	+1 Draw lines of constant X 0 Do not.

INTERNAL PARAMETERS
(continued)

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
IDRY	1	+1 Draw lines of constant Y 0 Do not.
IDRZ	0	+1 Draw lines of constant Z (contour lines) 0 Do not.
IUPPER	0	+1 Draw upper side of surface. 0 Draw both sides. -1 Draw lower side.
ISKIRT	0	+1 Draw a skirt around the surface. Bottom = HSKIRT. 0 Do not.
NCLA	6	Approximate number of levels of constant Z that are drawn if levels are not specified. 40 levels maximum.
THETA	.02	Angle in radians between eyes for STEREO pairs.
HSKIRT	0.	Height of skirt (if ISKIRT=1).
CHI	0.	Highest level of constant Z.
CLO	0.	Lowest level of constant Z.
CINC	0.	Increment between levels

} If CHI, CLO, or CINC is zero, a nice value is generated automatically.

A sample picture follows:



CALL EZSRFC (Z,40,30,220.,20.,WORK)

SUPMAP

**SUBROUTINE SUPMAP (JPROJ,POLAT,POLONG,ROT,PL1,PL2,PL3,PL4,JLTS,JGRID,
IUSOUT,IDOT,IER)**

DIMENSION OF ARGUMENTS PL1(2),PL2(2),PL3(3),PL4(2)

LATEST REVISION October, 1973

PURPOSE To plot continental and/or U.S. state outlines according to one of nine projections. The origin and orientation of the projection are selected by the user. Points on the earth defined by latitude and longitude are transformed to points in the U,V plane, the plane of projection. The U and V axes are respectively parallel to the X and Y axes of the dd80. A rectangular frame parallel to the U and V axes is chosen and only material within the frame is plotted.

ACCESS CARDS *FORTRAN,S=ULIB,N=SUPMAP
*COSY

USAGE If the following assumptions are met, use

CALL EZMAP (JPROJ,POLAT,POLONG)

Assumptions:

The Lambert conformal conic with two standard parallels, (IABS(JPROJ) = 3), is not required.

The maximum useful area for the projection is plotted.

USAGE
(continued)

Assumptions (continued)

The origin is at the point with latitude POLAT and longitude POLONG.

At the origin 'north' is parallel to the V axis.

Continental outlines are plotted as continuous lines.

Grids are plotted at 10° intervals.

The SUPMAP call is neither printed nor written beneath the map.

If these assumptions are not satisfied, use

CALL SUPMAP (JPROJ,POLAT,POLONG,ROT,PL1,PL2,PL3,PL4,JLTS,
JGRID,IUSOUT,IDOT,IER)

**On Input
for EZMAP**

JPROJ,POLAT,POLONG

See below under "On Input for SUPMAP."

**On Output
for EZMAP**

All arguments are unchanged.

**On Input
for SUPMAP**

JPROJ

IABS(JPROJ) defines the projection type according to the following code:

- 1 Stereographic
- 2 Orthographic
- 3 Lambert conformal conic with two standard parallels
- 4 Lambert equal area
- 5 Gnomonic
- 6 Azimuthal equidistant
- 7 Dummy--this code is not used
- 8 Cylindrical equidistant
- 9 Mercator
- 10 Mollweide type

If JPROJ is negative, the continental outlines are omitted.

**On Input
for SUPMAP**

(continued)

POLAT,POLONG,ROT

If (IABS(JPROJ).NE.3)

- POLAT and POLONG define in degrees the latitude and longitude of the point on the globe which is to transform to the origin of the U,V plane, where

$$-90 \leq \text{POLAT} \leq 90$$

$$-180 \leq \text{POLONG} \leq 180$$

Degrees of latitude north of the equator and degrees of longitude east of the Greenwich meridian are positive.

- ROT is the angle between the V axis and 'north' at the origin. It is measured in degrees and is taken to be positive if the angular movement from 'north' to the V axis is counter-clockwise. If the origin is at the north pole, 'north' is considered to be in the direction of (POLONG+180.). If the origin is at the south pole, 'north' is in the direction of POLONG. For the cylindrical projections (8,9,10), the axis of the projection is parallel to the V axis.

If (IABS(JPROJ).EQ.3) (Lambert conformal conic with two standard parallels)

- POLONG = central meridian of projection in degrees.
- POLAT,ROT are the two standard parallels in degrees.

JLTS,PL1,PL2,PL3,PL4

IABS(JLTS) can take the values 1 through 5 and specifies one of five options on the way in which the limits of the rectangular map are defined by the parameters

PL1,PL2,PL3,PL4.

IABS(JLTS) = 1

The maximum useful area produced by the projection is plotted. PL1, PL2, PL3, PL4 are not used and may be set to zero.

On Input
for SUPMAP
(continued)

IABS(JLTS) = 2

In this case PL1 through PL4 are the latitudes and longitudes in degrees of two points which are to be at opposite corners of the map. PL1, PL2 are latitude and longitude of a point at bottom left corner of map. PL3, PL4 are latitude and longitude of a point at top right corner.

IABS(JLTS) = 3

The minimum and maximum values of U and V are specified by PL1 through PL4. PL1 = UMIN, PL2 = UMAX, PL3 = VMIN, PL4 = VMAX. Knowledge of the transformation equations is necessary for this option to be used (see below).

IABS(JLTS) = 4

Here PL1 = AUMIN, PL2 = AUMAX, PL3 = AVMIN, PL4 = AVMAX, where

AUMIN = angular distance from origin to left frame of map.

AUMAX = angular distance from origin to right frame of map.

AVMIN = angular distance from origin to lower frame.

AVMAX = angular distance from origin to upper frame.

AUMIN, AUMAX, AVMIN, AVMAX must be positive and the origin must be within the rectangular limits of the map. This option is useful for polar projections. It is not appropriate for the Lambert conformal with two standard parallels. An error message is printed if an attempt is made to use JLTS = 4 when JPROJ = 3, (see below).

IABS(JLTS) = 5

PL1 through PL4 are two element arrays giving the latitudes and longitudes of four points which are to be on the four sides of the rectangular frame. PL1(1), PL1(2) are respectively the latitude and longitude of a point on the left frame. Similarly PL2 lies on the right frame, PL3 lies on the lower frame and PL4 lies on the upper frame. Note that in the calling program PL1 through PL4 will be dimensioned:

DIMENSION PL1(2), PL2(2), PL3(2), PL4(2)

If JLTS is positive, the SUPMAP call is written below the map. This is omitted if JLTS is negative.

**On Input
for SUPMAP
(continued)**

JGRID

IABS (JGRID) gives in degrees the interval at which lines of latitude and longitude are to be plotted. A value in the range 1 through 10 will usually be appropriate but higher values are acceptable. If JGRID

- < 0 The border around the map is omitted.
- = 0 No grid lines are plotted.
- = -0 Both grid and border are omitted.

IUSOUT

IABS(IUSOUT)

- = 1 U.S. state outlines are plotted.
- = 0 U.S. state outlines are not plotted.

Note that if U.S. state outlines are required, it will be usual to suppress the continental outlines by making JPROJ negative.

If IUSOUT is positive, the SUPMAP call and values of UMIN, UMAX, VMIN, VMAX are printed as an aid to debugging. This is omitted if IUSOUT = -0 or -1.

IDOT

- = 0 For continuous outlines.
- = 1 For dotted outlines.

**On Output
for SUPMAP**

All arguments except IER are unchanged.

IER

Error flag with the following meanings. If IER

- = 0 Map successfully plotted.
- = 33 Attempt to use non-existent projection.
- = 34 Map limits inappropriate.
- = 35 Angular limits too great.
- = 36 Map has zero area.
- = 37-40 Failures in ULLIB data access. If IER
 - = 37 EOF on ULLIB file.
 - = 39 Unsuccessful ULLIB read.
 - = 40 No check on last ULLIB operation.

ENTRY POINTS

EZMAP,MAPLOT,SUPCON,SUPCONQ,SUPFST,SUPMAP,SUPTRP,SUPVEC,
SUPVECQ,VECPLT

MAPLOT

Actually draws the map.

SUPCON

Once the transformation has been set up by an initial call to SUPMAP, the subroutine SUPCON may be called to transform a point, (latitude, longitude) to the corresponding point, (U,V) on the plane. Contours may thus be readily drawn against the map background.

CALL SUPCON(RLAT,RLON,U,V)

On Input:

RLAT,RLON are the latitude and longitude of a point to be transformed to the U,V plane. $-90. \leq \text{RLAT} \leq 90.$
 $-180. \leq \text{RLON} \leq 180.$

On Output:

RLAT,RLON are unchanged.
U,V are the transformed coordinates of the point (RLAT,RLON).

SUPCONQ

Actually performs the above mentioned transformation.

SUPFST

SUPVEC

To facilitate drawing lines on the map these routines which act like the plotting routines FRSTPT and VECTOR are included. They are subject to the same restrictions as SUPCON above.

CALL SUPFST (RLAT,RLON)

CALL SUPVEC (RLAT,RLON)

SUPVECQ

The routine which actually draws most of the lines.
(C.F. MAPLOT and VECPLT)

ENTRY POINTS
(continued)

SUPTRP

Performs interpolation to the edges of the frame.

VECPLT

Called by SUPVECQ.

COMMON BLOCKS

SUPMP1 of length 4810 = 608

SUPMP2 of length 102510 = 20018

(Occurs only in SUPMAP and MAPLOT. Blank common may be used.)

I/O

Map plotted on dd80. Outline data read from ULIB. SUPMAP call printed (possibly).

PRECISION

Single

**REQUIRED ULIB
ROUTINES**

ULIBER (on system).

SPECIALIST

Cicely Ridley and J.W. Chalmers, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORYRevised January 1969, May 1971, standardized October 1973;
revised July 1974.**ALGORITHM**

The latitudes and longitudes of successive outline points are transformed to coordinates in the plane of projection and joined by a vector.

REFERENCES

Hershey, A.V., The Plotting of Maps on a CRT Printer. NWL Report No. 1844, 1963.

Lee, Tso-Hwa, Students' Summary Reports, Work-Study Program in Scientific Computing. NCAR 1968.

Parker, R.L., 2UCSD SUPERMAP: World Plotting Package.

Steers, J.A., An Introduction to the Study of Map Projections. University of London Press, 1962.

SPACE REQUIRED

Approximately 5000, (not including plotting routines and common blocks.)

ACCURACY

The definition of the map produced is limited by two factors:

- The outline data has a resolution of 1°.
- The resolution of the dd80 is limited to 1024 units in the X and Y directions.

TIMING

Usually less than one second per map depending upon projection, origin and orientation. The cylindrical equidistant projection with POLAT = ROT = 0.0 is particularly fast.

PORTABILITY

This package involves dd80 plotting and is therefore not highly portable. However, plotting instructions are largely concentrated in the routines MAPLOT, SUPVECQ and VECPLT. The transformation routines SUPCON and SUPCONQ are portable.

PLOTTING ROUTINES USED

PWRT, FRSTPT, VECTOR, POINT, DASHLN, PERIM, SET

REQUIRED RESIDENT ROUTINES

ATAN, TAN, SIN, COS, ALOG, SQRT, ATAN2, ACOS

More History

R. L. Parker of UCSD wrote the original SUPERMAP system. This was adapted for use on the NCAR system by Lee (1968). The present SUPMAP is an improved version. The information for plotting continental and state outlines is due to Hershey (1963). The nine projections incorporated in the routine are described briefly below. For more detail refer to Stears (1962).

Mathematical Method

The ULIB data file, SUPMAPD, contains the latitudes and longitudes of points on the continental and U.S. state outlines at a resolution of about 1° . The latitude and longitude of successive points on the outline are transformed to coordinates in the U,V plane and joined by a vector. To improve definition, points closer than 4 CRT units are discarded. The parameter IDOT enables a dotted outline to be selected. In this case, points are linearly inserted if successive data points are separated by more than 4 CRT units. A similar procedure is used to plot the grid lines joining successive points by a dotted vector.

The subroutine SUPMAP calculates the limits of the rectangular map before calling subroutine MAPLOT to plot the continental outlines, grid lines and frame. Both SUPMAP and MAPLOT call subroutine SUPCON to transform a point on the globe to a point in the U,V plane.

Description of Projections Used

Asimuthal Projections (IABS(JPROJ) = 1,2,4,5,6)

The U,V plane is tangential to the globe at the point $\phi(\text{POLAT},\text{POLONG})$, which transforms to the origin, $\phi'(0,0)$, of the U,V plane (see Figure 1). ROT is the angle between the V-axis and north at ϕ' .

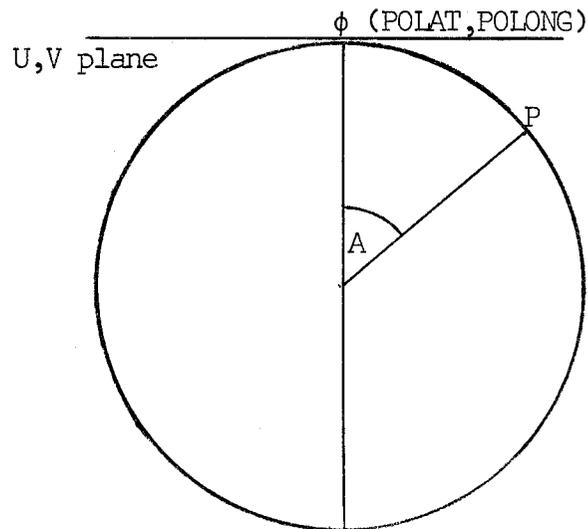


Figure 1

Let P be a point on the globe at an angular distance, A, from the point $\phi(\text{POLAT},\text{POLONG})$. Let B be the angle between the great circle ϕP and the meridian at ϕ . Let P transform to P' in the U,V plane (see Figure 2). The program calculates $\text{SINA} = \text{SIN}(A)$, $\text{COSA} = \text{COS}(A)$, $\text{SINB} = \text{SIN}(B)$, $\text{COSB} = \text{COS}(B)$ in terms of the latitude and longitudes of ϕ and P. Then, if the distance $\phi'P'$ is R and $\text{SINR} = \text{SIN}(\text{ROT})$, $\text{COSR} = \text{COS}(\text{ROT})$, point P' has coordinates

$$\begin{aligned} U &= R \cdot \text{SIN}(B + \text{ROT}) = R \cdot (\text{SINB} \cdot \text{COSR} + \text{COSB} \cdot \text{SINR}) \\ V &= R \cdot \text{COS}(B + \text{ROT}) = R \cdot (\text{COSB} \cdot \text{COSR} - \text{SINB} \cdot \text{SINR}) \end{aligned}$$

It remains to define R in terms of COSA and SINA.

Description of
Projections Used
(continued)

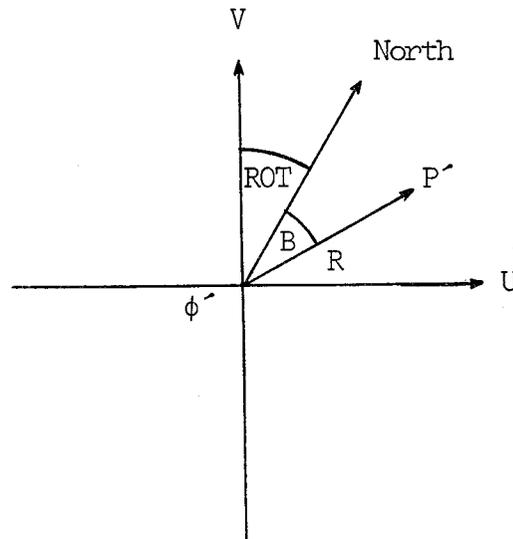


Figure 2

Stereographic Projection (1)

$$R = \text{TAN}(A/2) = (1.-\text{COSA})/\text{SINA} \quad \text{as } A \rightarrow 180^{\circ}, R \rightarrow \infty.$$

Thus the entire surface of the globe transforms to the entire U,V plane. In practice distortion becomes great beyond $R = 2$, or $A \sim 130^{\circ}$.

Orthographic Projection (2)

$$R = \text{SINA}$$

This projection plots a hemisphere within radius $R = 1$. The maximum possible value of $A = 90^{\circ}$.

**Description of
Projections Used**
(continued)

Lambert Equal Area Projection (4)

R is calculated by the Fortran statement

$$R = 2.*SINA/SQRT(2.*(1.+COSA))$$

As $A \rightarrow 180^\circ$, $R \rightarrow 2$, and the entire surface of the globe is plotted within a radius $R = 2$. The maximum value of $A = 180^\circ$.

In this projection the area between two circles, center ϕ' , is proportional to the corresponding area on the globe.

Gnomonic Projection (5)

$$R = SINA/COSA \quad \text{As } A \rightarrow 90^\circ, R \rightarrow \infty.$$

A hemisphere is plotted over the entire U,V plane. In practice, distortion becomes great beyond $R = 2$, or $A \sim 65^\circ$.

Azimuthal Equidistance Projection (6)

$$\begin{aligned} R &= A \text{ (in radians)} \\ &= ACOS(COSA) \quad \text{As } A \rightarrow 180^\circ, R \rightarrow \pi. \end{aligned}$$

The entire globe surface is plotted within a radius $R = \pi$.

Description of
Projections Used
(continued)

Cylindrical Projections (IABS(JPROJ) = 8,9,10)

The U,V plane must be imagined to be wrapped around the globe to form a cylinder, the U-axis touching the globe on some great circle (see Figure 3). The axis of the projection is perpendicular to this great circle and parallel to the V-axis. The point ϕ (POLAT,POLONG), which transforms to the origin, $\phi'(0,0)$, of the projection, lies on the great circle. The limits of the U-axis are defined by a cut in the cylinder parallel to its axis and diametrically opposite to ϕ . The pole of the projection, Q, is the point 90° from the great circle in the direction of +V. ROT is the angle between the V-axis and north at ϕ . These points and N, the north pole, are shown in Figure 3. Points on the surface of the globe are transformed to points on the U,V cylinder by the rule appropriate to the projection.

The latitude and longitude of Q are calculated in terms of the latitude and longitude of ϕ and the angle ROT. The angle ROT1 (see Figure 3) is also computed.

In Figure 3, P is some general point on the surface of the globe. A is the angular distance of P from Q. B is the angle between the great circles QN and QP. The quantities

$$\begin{array}{ll} \text{SINA} = \text{SIN}(A), & \text{COSA} = \text{COS}(A) \\ \text{SINB} = \text{SIN}(B), & \text{COSB} = \text{COS}(B) \\ \text{SINR} = \text{SIN}(\text{ROT1}), & \text{COSR} = \text{COS}(\text{ROT1}) \end{array}$$

are computed.

Let $P'(U,V)$ be the point on the U,V plane corresponding to the point P on the surface of the globe. Then U is proportional to the angle α (see Figure 3). The value of V depends upon the type of projection. The coordinates of P' are given below.

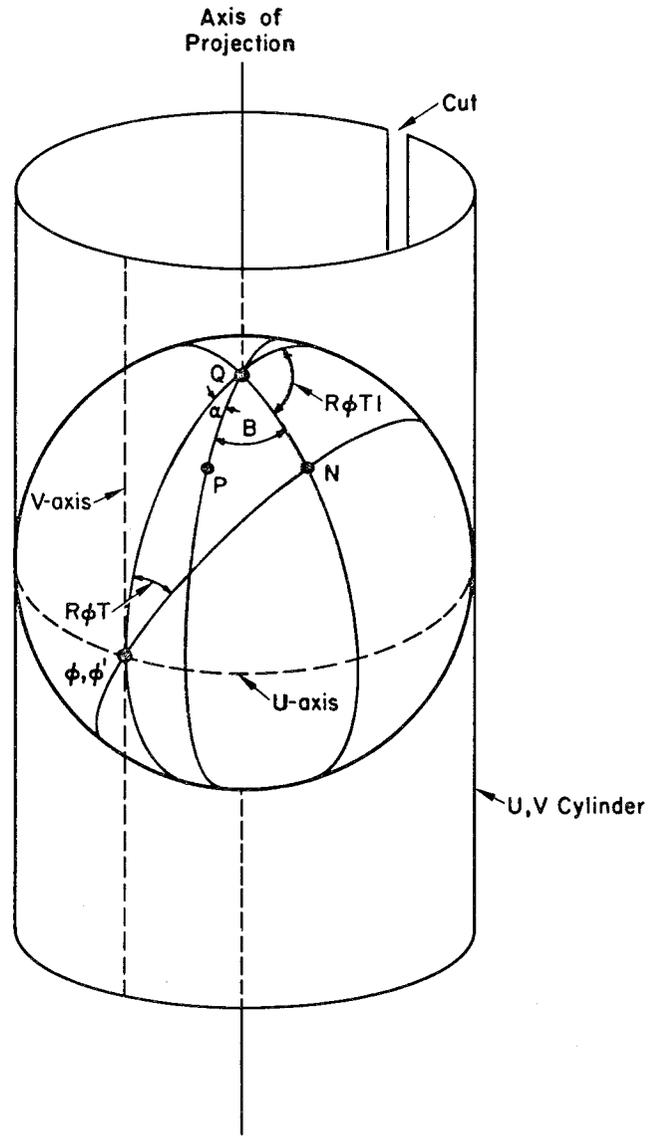


Figure 3

Description of
Projections Used
(continued)

Cylindrical Equidistant Projection (8)

$$\begin{aligned} U &= \alpha \text{ (in degrees)} \\ &= \text{ATAN2}(\text{SIN}(B+\text{ROT1}), -\text{COS}(B+\text{ROT1}))/F \\ &= \text{ATAN2}(\text{SINB}*\text{COSR}+\text{COSB}*\text{SINR}, \text{SINB}*\text{SINR}-\text{COSB}*\text{COSR})/F \end{aligned}$$

Here division by F converts radians to degrees.

$$\begin{aligned} V &= 90.-A \text{ (in degrees)} \\ &= 90.-\text{ACOS}(\text{COSA})/F \end{aligned}$$

The entire surface of the globe is transformed to a rectangle in the U,V plane.

$$\begin{aligned} -180. &\leq U \leq 180. \\ -90. &\leq V \leq 90. \end{aligned}$$

Mercator Projection with Arbitrary Pole (9)

$$\begin{aligned} U &= \alpha \text{ (in radians)} \\ &= \text{ATAN2}(\text{SINB}*\text{COSR}+\text{COSB}*\text{SINR}, \text{SINB}*\text{SINR}-\text{COSB}*\text{COSR}) \\ V &= \text{ALOG}(\text{COT}(A/2)) \\ &= \text{ALOG}((1.+\text{COSA})/\text{SINA}) \end{aligned}$$

The entire surface of the globe is transformed to an infinite rectangle in the U,V plane. As $A \rightarrow 0$, $V \rightarrow \infty$, as $A \rightarrow 180^\circ$, $V \rightarrow -\infty$. When $\alpha = 180^\circ$, $U = \pi$, when $\alpha = -180^\circ$, $U = -\pi$.

Hence

$$\begin{aligned} -\infty &\leq V \leq \infty \\ -\pi &\leq U \leq \pi \end{aligned}$$

In practice distortion becomes great for $A < 5^\circ$ or $> 175^\circ$.

Description of
Projections Used
(continued)

Mollweide-Type Projection (10)

The projection used is not a true Mollweide. The coordinates of P' are given by

$$V = \cos A$$

U is given by the two Fortran statements

$$U = \text{ATAN2}(\text{SINB}*\text{COSR}+\text{COSB}*\text{SINR}, \text{SINB}*\text{SINR}-\text{COSB}*\text{COSR})*\text{SF}$$

where $\text{SF} = 2./\pi$

$$U = U*\text{SQRT}(1-V*V)$$

The entire surface of the globe transforms to an ellipse in the U,V plane. The major and minor axes of the ellipse are along the U and V axes respectively.

$$-2. \leq U \leq 2.$$

$$-1. \leq V \leq 1.$$

**Description of
Projections Used**
(continued)

Conical Projections (IABS(JPROJ) = 3)

In a conical projection, the meridians are represented by straight lines radiating from the apex of the flattened cone. Any two longitudes ϕ_1 and ϕ_2 transform to two straight lines at an angle $n(\phi_2 - \phi_1)$, where n is the cone constant, less than unity. The parallels transform to circles centered at the apex. The radius of the circle corresponding to latitude and is given by

$$R = f(\theta) \quad .$$

The cone constant, n , and the function f depend upon the type of conical projection in use.

In Figure 4, the longitudes ϕ_1, ϕ_2 are represented by PA, PB. Then

$$\angle APB = n(\phi_2 - \phi_1)$$

CD is the arc of a circle center P, radius R and represents latitude θ . Then

$$R = f(\theta) \quad .$$

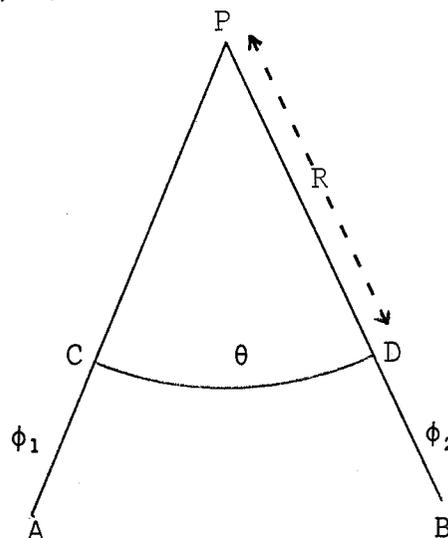


Figure 4

**Description of
Projections Used**
(continued)

Lambert Conformal with Two Standard Parallels (3)

This projection has the property of preserving angles and is relatively distortion free for mid-latitude regions. The cone constant is given by

$$n = \frac{\log(\cos\theta_1) - \log(\cos\theta_2)}{\log(\tan(45 \pm \frac{\theta_1}{2})) - \log(\tan(45 \pm \frac{\theta_2}{2}))}$$

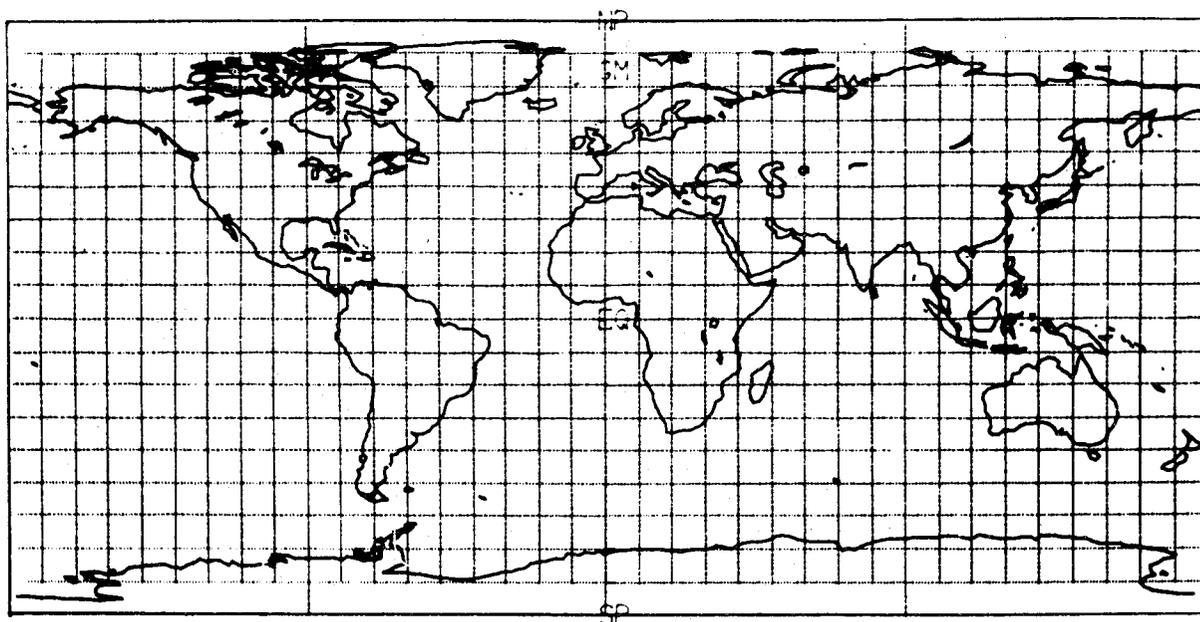
where the lower signs are for the northern hemisphere, upper signs the southern. The relationship between R and is

$$R = [\tan(45 \pm \frac{\theta}{2})]^n$$

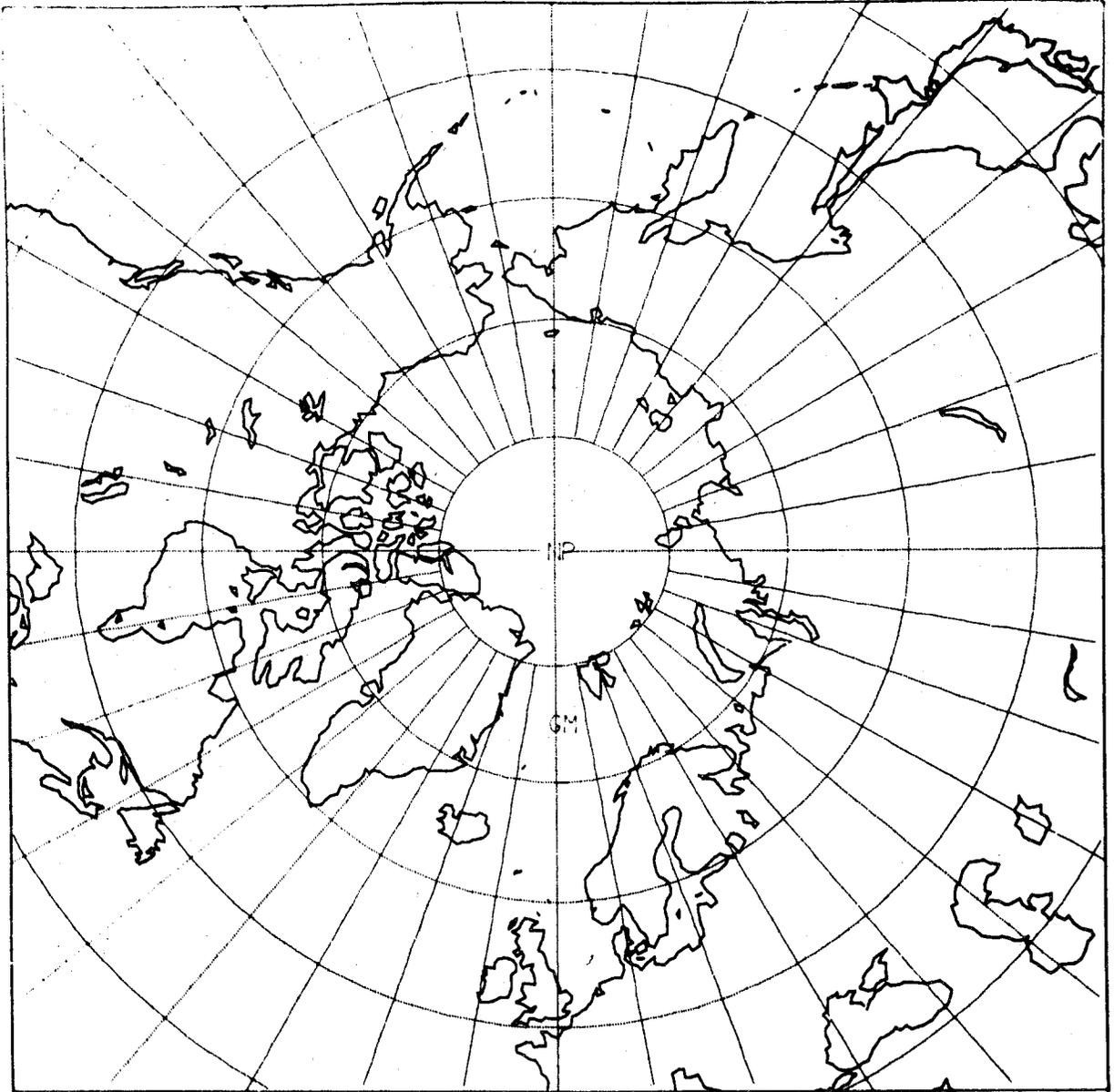
The apex of the cone is the origin of the (U,V) plane of projection. The central meridian, ϕ_0 , is parallel to the V axis with north in the positive direction. Then

$$U = R \sin[n(\phi - \phi_0)]$$

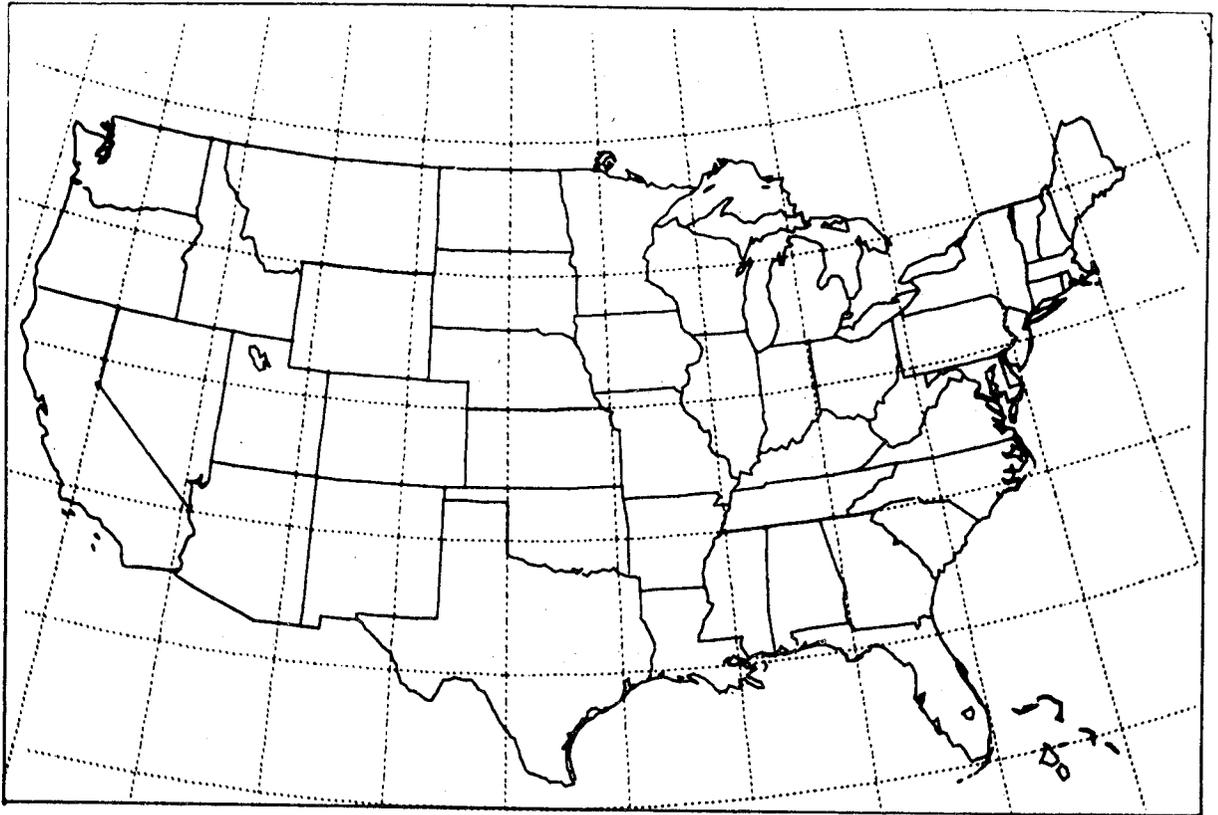
$$V = \pm R \cos[n(\phi - \phi_0)]$$



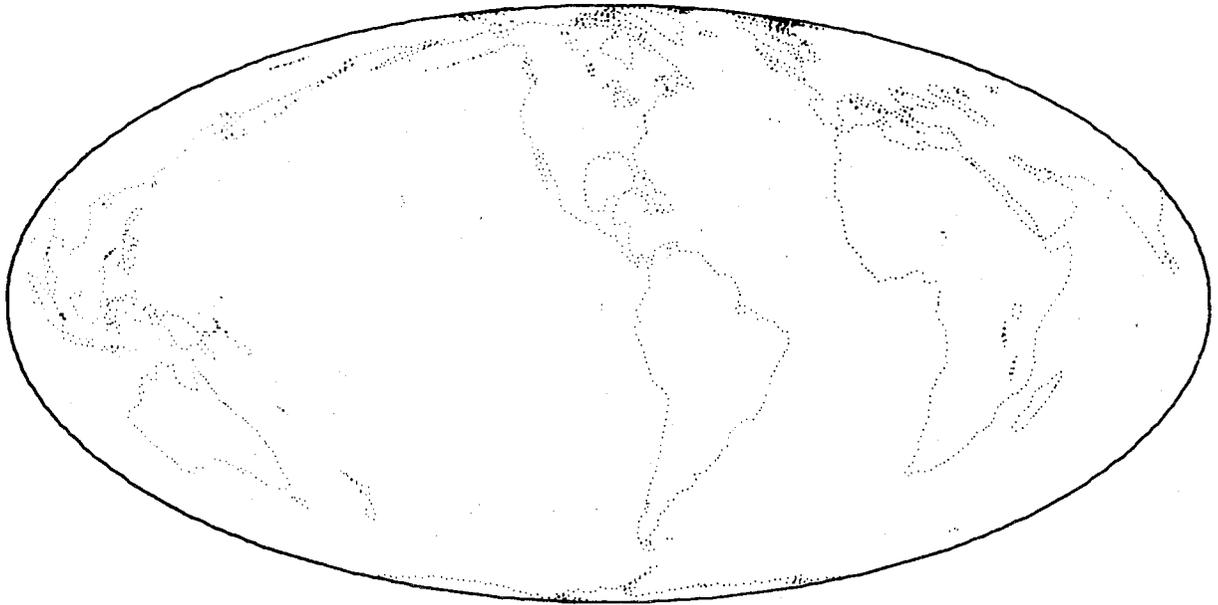
SUPMAP: 8, 0.0, 0.0, 0.0, -90.0, -180.0, 90.0, 180.0, 2, 10, 0, 0;



SUPMAP(1, 90.0, 0.0, 0.0, 45.0, 45.0, 45.0, 45.0, 4, 10, 0, 0)



SUPMAP(-5, 30.0,-100.0, 50.0, 40.0, 40.0, 25.0, 55.0, 5, 5, 1, 0;
-125.0, -65.0,-100.0,-100.0,



SUPMAP(10, 0.0, -90.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1, -0, 0, 1)

SUBROUTINE VELVEC (U,LU,V,LV,M,N,FLO,HI,NSET,ISPV,SPV)**DIMENSION OF
ARGUMENTS**

U(LU,N),V(LV,N),SPV(2)

LATEST REVISION

November 1973

PURPOSE

VELVEC draws a representation of a two dimensional velocity field by drawing arrows from each data location, with the length of the arrow proportional to the strength of the field at that location and the direction of the arrow indicating the direction of the flow at that location.

ACCESS CARDS

*FORTRAN,S=ULIB,N=VELVEC
*COSY

USAGE

If the following assumptions are met, use
CALL EZVEC (U,V,M,N)

Assumptions:

The whole array is to be processed.

The scale factor is chosen internally.

USAGE

(continued)

The perimeter is drawn by EZVEC.

FRAME is called by EZVEC.

There are no special values.

If these assumptions are not met, use

CALL VELVEC (U,LU,V,LV,M,N,FLO,HI,NSET,ISPV,SPV)

ARGUMENTS

On Input

U,V

The origins of the two dimensional arrays containing the velocity field to be plotted. The point (I,J) having magnitude $\text{SQRT}(U(I,J)**2+V(I,J)**2)$ and direction $\text{ATAN2}(V(I,J),U(I,J))$.

LU

The first dimension of U in the calling program.

LV

The first dimension of V in the calling program.

M,N

The number of data values to be plotted in the X-direction (the first subscript direction), and Y-direction (the second subscript direction). When plotting the entire array, $LU=LV=M$. See Appendix 1 of this chapter for an explanation of using this argument list to process any part of an array.

FLO

The minimum vector length to be plotted.

HI

The **maximum** vector length to be plotted. (If=0, the maximum of the array will be chosen.)

On Input
(continued)

NSET

Flag to control scaling.

- = 0 VELVEC calls SET to properly scale the plotting instructions to the standard configuration. PERIM is called to draw a border.
- > 0 If NSET is positive, VELVEC assumes that SET has been called by the user in such a way as to properly scale the plotting instructions generated by VELVEC. PERIM is not called.
- < 0 If NSET is negative, VELVEC calls SET in such a way as to place the contour plot within the limits of the user's last SET call. PERIM is not called.

ISPV

Flag to control the special value feature.

- = 0 Means that the feature is not in use.
- = 1 Means that if the value of $U(I,J)=SPV(1)$ the vector will not be plotted.
- = 2 Means that if the value of $V(I,J)=SPV(2)$ the vector will not be plotted.
- = 3 Means that if either $U(I,J)=SPV(1)$ or $V(I,J)=SPV(2)$ then the vector will not be plotted.
- = 4 Is identical to ISPV=3 with $SPV(2)=SPV(1)$.

On Output

All arguments remain unchanged.

NOTE

The endpoints of the vector are computed by:

$$(FX(X,Y),FY(X,Y))$$

and

$$(FX2(X,Y,U,V,SF),FY2(X,Y,U,V,SF))$$

where $X = I$, $Y = J$, $U = U(I,J)$, $V = V(I,J)$, and $SF = \text{scale factor}$. Here $I = X\text{-index}$, $J = Y\text{-index}$. Thus the actual magnitude of the vector is $\text{SQRT}(DX^{**2}+DY^{**2})$ and the direction is $\text{ATAN2}(DY,DX)$, where $DX = FX(\dots)-FX2(\dots)$ and $DY = FY(\dots)-FY2(\dots)$.

When user defined transformations are used, be sure to include definitions for $FX2$ and $FY2$. Usually the definitions:

$$FX2(X,Y,U,V,SF) = FX(X+U*SF,Y+V*SF)$$

and

$$FY2(X,Y,U,V,SF) = FY(X+U*SF,Y+V*SF) \text{ will suffice.}$$

See Appendix 2 of this chapter for more details.

ENTRY POINTS

VELVEC, EZVEC, DRWVEC

COMMON BLOCKS

VECL 16₈

I/O

Plots the vector field.

PRECISION Single

REQUIRED ULIB ROUTINES None

SPECIALIST Jay W. Chalmers, NCAR, Boulder, Colorado 80303

LANGUAGE FORTRAN

HISTORY Written and standardized in November 1973.

ALGORITHM Each vector is examined and possibly transformed, then plotted.

SPACE REQUIRED Approximately 1300₈, not including the system plot package.

PORTABILITY An implementor's write-up is available.

PLOTTING ROUTINES REQUIRED SET, GETSET, PERIM, FRSTPT, VECTOR, MXY

REQUIRED RESIDENT ROUTINES SQRT, ATAN2, SIN, COS, ENCD, PWR

INTERNAL PARAMETERS	<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
	BIG	-1.E320	Constant used to initialize possible search for HI.
	EXT	0.25	Lengths of the sides of the plot are proportional to M and N when NSET \leq 0, except in the case when $\text{MIN}(M,N)/\text{MAX}(M,N) < \text{EXT}$, in which case a square graph is plotted.
	ICTRFG	1	Flag to control the position of the vector relative the base point. \equiv 0 Center at (MX,MY) $>$ 0 Tail at (MX,MY) $<$ 0 Head at (MX,MY)
	ILAB	0	Flag to control the drawing of line labels. \equiv 0 Do not draw the labels \neq 0 Draw the labels
	INCX	1	X-coordinate step size for less dense arrays.
	INCY	1	Y-coordinate step size.
	IOFFD	0	Flag to control normalization of label numbers. \equiv 0 Means include a decimal point when possible. (Do not normalize.) \neq 0 Means normalize all label numbers by ASH.
	IOFFM	0	Flag to control plotting of the message below the plot. \equiv 0 Means plot the message \neq 0 Means do not plot it.
	RMN	5.0	Minimum size line length to scale vector. (CRT units.)
	RMX	200.0	Maximum size line length to scale vector. (CRT units)
	SIDE	0.90	Length of longer edge of plot. (See also EXT.)
	SIZEP	1.25	Size of FWRY labels for vector values.
	XLT	0.05	Left hand edge of the plot. (0.0 \equiv left edge of the frame. 1.0 \equiv right hand edge.)

INTERNAL PARAMETERS
(continued)

<u>NAME</u>	<u>DEFAULT</u>	<u>FUNCTION</u>
YBT	0.05	Bottom edge of the plot (0.0 \equiv bottom of frame 1.0 \equiv top of frame.)
ZMN	FLO	Minimum line length to plot. (CRT units.)
ZMX	0.0	Maximum line length to generate a degenerate vector. (A point.)

Internal functions which may be modified for transformation
of the data:

SCALE

Computes a scale factor used in the determination of the
length of the vector to be drawn.

DIST

Computes the length of a vector.

FX

Returns the X index as the X-coordinate of the vector base.

FX2

Returns the X-coordinate of the vector head.

FY

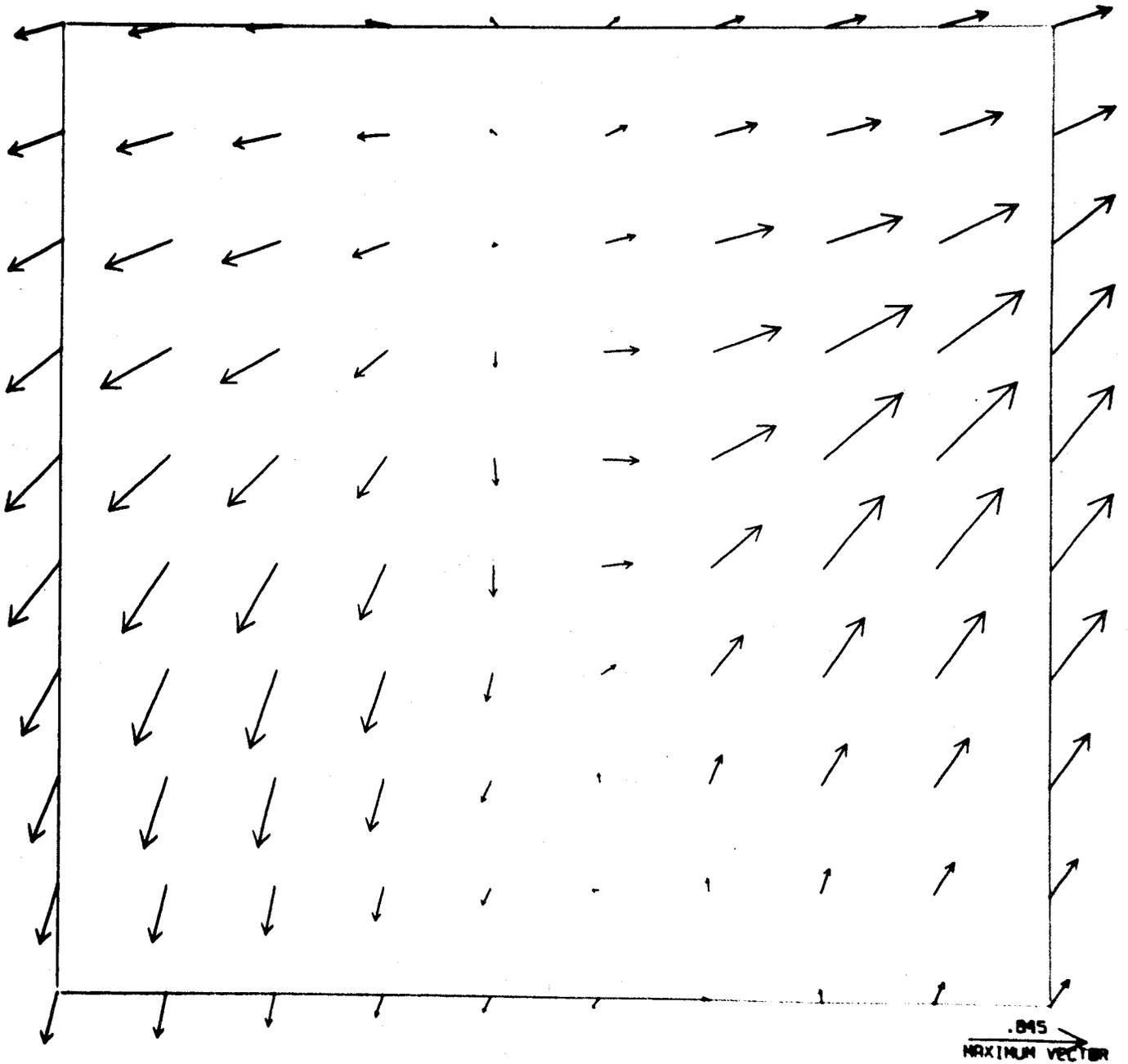
Returns the Y index as the Y-coordinate of the vector base.

FY2

Returns the Y-coordinate of the vector head.

VLAB

The value for the vector label when ILAB \neq 0.



APPENDIX 1: PROCESSING OF PARTS OF ARRAYS

INTRODUCTION

The standards for argument lists of functions and subroutines in ULIB require a set of arguments (described below) which make it possible to process not only the entire array, but any part (sub-array) from the array.

REQUIRED ARGUMENTS

When an N dimensional array is being passed to a subroutine, the following arguments are required: The name (origin) of the array, its first N-1 dimensions in the dimension statement of the calling routine, and the number of array values to be processed in each of the N dimensions. (It is not necessary to pass all N dimensions from the dimension statement of the calling program because only the first N-1 are used in the addressing algorithm.)

Example 1: Processing all of an array

```
PROGRAM MAIN
DIMENSION A(100),B(30,20),C(22,16,9)
C THESE CALLS PROCESS ALL OF EACH ARRAY
CALL LINEAR (A,100)
CALL TWODIM (B,30,30,20)
CALL THREED (C,22,22,16,16,9)
STOP
END
SUBROUTINE LINEAR (U,N)
DIMENSION U(N)
DO 1 I=1,N
(Process U array)
1 CONTINUE
RETURN
END
SUBROUTINE TWODIM (V,LX,MX,MY)
DIMENSION V(LX,MY)
DO 1 I=1,MX
DO 1 J=1,MY
(Process V array)
1 CONTINUE
RETURN
END
SUBROUTINE THREED (W,LX,MX,LY,MY,MZ)
DIMENSION W(LX,LY,MZ)
DO 1 I=1,MX
DO 1 J=1,MY
DO 1 K=1,MZ
(Process W array)
1 CONTINUE
RETURN
END
```

**PROCESSING PART
OF AN ARRAY**

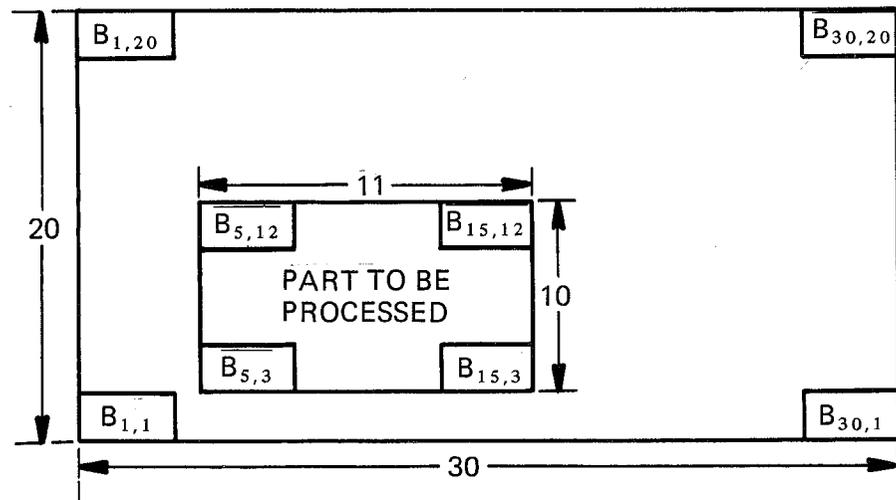
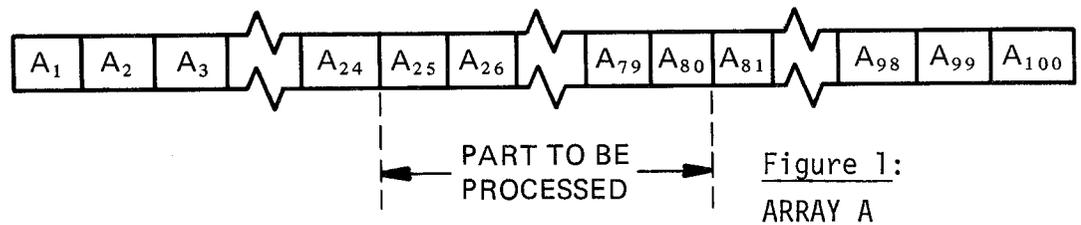
These same arguments allow a user to process any array contained within another array. When the user is doing this, the array argument can be used as the origin of the sub-array. The following examples demonstrate the processing of parts of arrays (and assume the same subroutines as listed in Example 1).

Example 2: Processing part of an array

```

PROGRAM MAIN
  DIMENSION A(100),B(30,20),C(22,16,9)
  C PROCESS FROM A(25) THROUGH A(80),} See Figure 1.
  C (80-25+1=56)
  CALL LINEAR (A(25),56)
  C PROCESS PART OF B STARTING AT B(5,3) AND
  C INCLUDING 11 B's IN THE FIRST SUBSCRIPT DIRECTION
  C AND 10 B's IN THE SECOND SUBSCRIPT DIRECTION.
  C (A TOTAL OF 11*10=120 B's.) See Figure 2.
  CALL TWODIM (B(5,3),30,11,10)
  C PROCESS A CUBE IN C WITH ONE CORNER HAVING
  C SUBSCRIPTS (7,4,2) AND THE OPPOSITE} See Figure 3.
  C CORNER AT (18,14,8)
  CALL THREED (C(7,4,2),22,12,16,11,7)
  STOP
  END
  
```

This concept is consistent with Standard FORTRAN and should be portable.



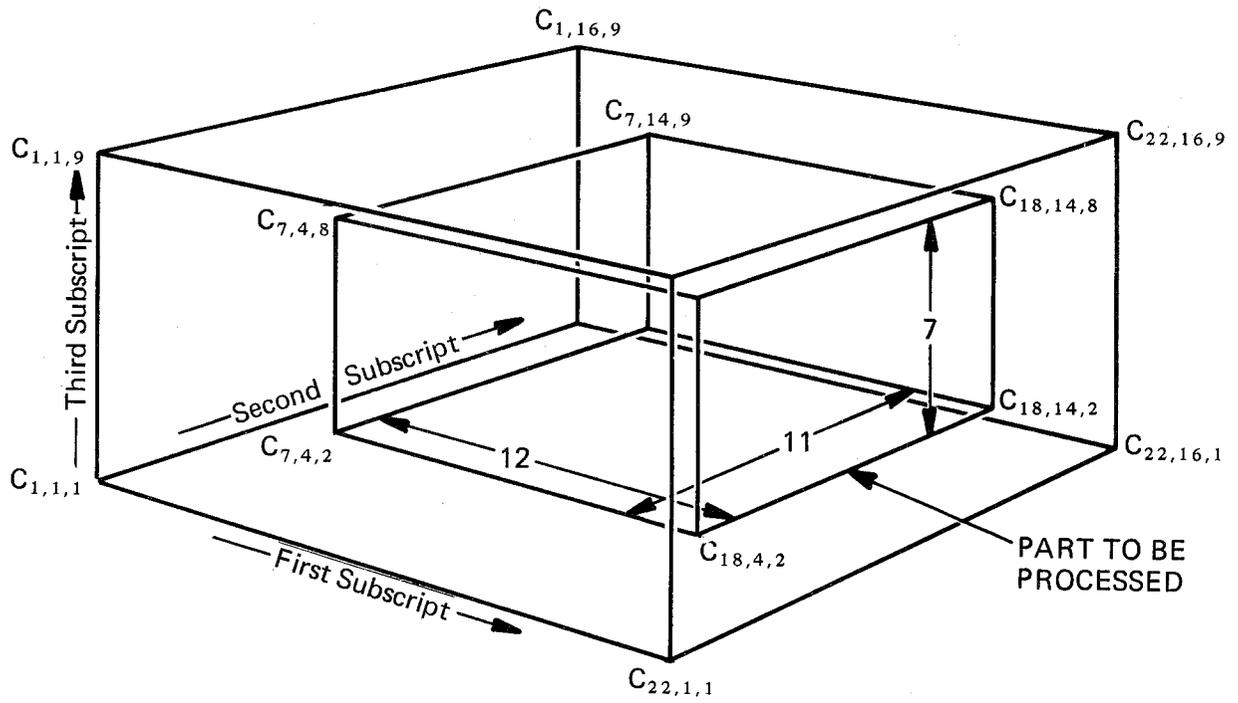


Figure 3:
ARRAY C

APPENDIX 2: TRANSFORMATIONS

INTRODUCTION

Where it is appropriate, the graphics routines contain the facility to transform the lines drawn from the default positions to other positions on the screen resulting in a new shape for the picture. For example, a contour map generated from data stored in a rectangular array in core can be transformed to produce a polar coordinate plot.

IMPLEMENTATION

In all packages with transformations, they are implemented by including statement functions containing the default transformations in the appropriate routines. (These transformations are very short and result in little or no run time overhead.¹) The user wanting other than the default transformations can replace the statement functions with new ones or, if the transformation is complicated, delete the statement functions and add external routines with the same name to do the transforming.

The range of the arguments to the statement functions is dependent on the routine in use. (Explanations of several of the more common transformations used in contouring and velocity vectors given below illustrate this point.) The

¹ There is zero run time overhead at NCAR.

IMPLEMENTATION
(continued)

range of the default transformation is the same as that of the arguments. If a user-supplied transformation has a different range, it must be considered. This is usually done by calling SET before calling the plotting routine and suppressing the SET call within the plotting routine via one of its arguments.

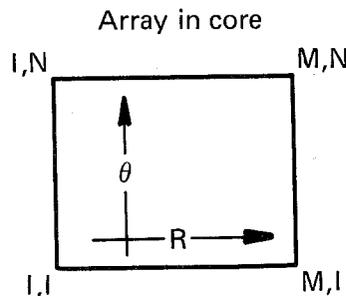
EXAMPLES

The following examples assume that the range of arguments to the statement functions is the range found in all the contouring routines for data stored in rectangular arrays and the velocity vector routine. In both types of routines, M by N arrays are supplied by the user. The default transformations are

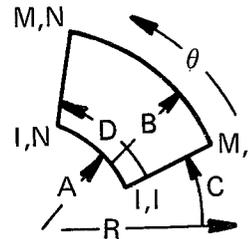
$$\begin{aligned} FX(X,Y) &= X \\ FY(X,Y) &= Y \end{aligned}$$

X corresponds to the first subscript direction. Y corresponds to the second subscript direction. $1 \leq X \leq M$ and $1 \leq Y \leq N$, but X and Y can take on non integer values. When $X = 1.0$ and $Y = 1.0$, the algorithm has generated a coordinate corresponding to the array value with subscripts (1,1).

Example 1: Polar Coordinates



Desired plot arrangement



Default transformations

$$\begin{aligned} FX(X,Y) &= X \\ FY(X,Y) &= Y \end{aligned}$$

Defaults are replaced by

```
COMMON /TRANS/A,B,C,D
FX(X,Y)=(A+B*(X-1.)/(FLOAT(M)-1.))*COS(C+D*(Y-1.)/(FLOAT(N)-1.))
FY(X,Y)=(A+B*(X-1.)/(FLOAT(M)-1.))*SIN(C+D*(Y-1.)/(FLOAT(N)-1.))
```

Note:

- A, B, C, and D are in common with the routing that called the plotting routine
- C and D are the desired angles in radians
- A and B are the desired distances
- Here and in the subsequent examples, SET was called by the the routine that called the plotting routine in such a way to accommodate the numbers generated by FX and FY. CALL SET(.1,.9,.1,.9,0,A+B,0.,A+B,1) would work in this case.

Example 2: Orthogonal unequally spaced

Z(I,J) is located at (XX(I),YY(J))

Default transformations are replaced by

```
COMMON /TRANS/XX(100),YY(100)
FX(X,Y)=XX(IFIX(X))+((XX(IFIX(X)+1)-XX(IFIX(X)))*(X-AINT(X)))
FY(X,Y)=YY(IFIX(Y))+((YY(IFIX(Y)+1)-YY(IFIX(Y)))*(Y-AINT(Y)))
```

EXAMPLES
(continued)

Note:

- XX(I) has been set to the proper abscissa for I=1,...,M.
- YY(J) has been set to the proper ordinate for J=1,...,N.
- XX(M+1) has been set equal to XX(M).
- YY(N+1) has been set equal to YY(N).

Example 3: Overlaying on SUPMAP

The default transformations are deleted and the following routines are added to the program:

```
FUNCTION FX(X,Y)
COMMON /TRANS/M,N,XLONMI,XLONMA,YLATMI,YLATMA,YANS
XLON=XLONMI+(XLONMA-XLONMI)*(X-1.)/(FLOAT(M)-1.)
YLAT=YLATMI+(YLATMA-YLATMI)*(Y-1.)/(FLOAT(N)-1.)
CALL SUPCON(YLAT,XLON,XANS,YANS)
FX=XANS
RETURN
END
FUNCTION FY(X,Y)
COMMON /TRANS/M,N,XLONMI,XLONMA,YLATMI,YLATMA,YANS
FY=YANS
RETURN
END
```

Note:

- SUPMAP must be called before CONREC or VELVEC
- The longitudes of the data are equally spaced between XLONMI and XLONMA.
- The latitudes of the data are equally spaced between YLATMI and YLATMA. Z(1,1) is at longitude XLONMI, latitude YLATMI.
- M,N,XLONMI,XLONMA,YLATMI, and YLATMA are initialized in the user's routine which calls the plotting routines.

Example 4: A parameterized distortion, 30 by 20 example

Z(I,J) is at (XX(I,J),YY(I,J))

The default transformations are deleted and the following routines are added to the program:

```

      FUNCTION FX(X,Y)
      COMMON /TRANS/XX(30,20),YY(30,20),YS
      I=X
      J=Y
      IF(FLOAT(I).EQ.X) GO TO 1
C Y=J
      FX=XX(I,J)+(XX(I+1,J)-XX(I,J))*(X-FLOAT(I))
      YS=YY(I,J)+(YY(I+1,J)-YY(I,J))*(X-FLOAT(I))
      RETURN
C TEST FOR CORNER
1 IF(FLOAT(J).EQ.Y) GO TO 2
      FX=XX(I,J)+(XX(I,J+1)-XX(I,J))*(Y-FLOAT(J))
      YS=YY(I,J)+(YY(I,J+1)-YY(I,J))*(Y-FLOAT(J))
      RETURN
2 FX=XX(I,J)
      YS=YY(I,J)
      RETURN
      END
      FUNCTION FY(X,Y)
      COMMON /TRANS/XX(30,20),YY(30,20),YS
      FY=YS
      RETURN
      END
  
```


13

UTILITY PROCESSORS

EDITOR

FIDEL

FLEX

FRED



PROGRAM EDITOR

LATEST REVISION February 1974

PURPOSE The program EDITOR allows the user to create and maintain a tape library of source deck files in a PLIB-compatible form.

Files read from the card reader (with sequence checking, if desired), from PLIB, or from an EDITOR-written input tape may be edited, resequenced, listed, and/or renamed, then punched, written to PLIB, or written to an output tape.

USAGE An EDITOR run deck has the following format:

```
*JOB,----,-----,-----
*LIMIT,T=--,PT=--,PR=--
*ASSIGN,-----=11,R (input tape--if none, leave card out)
*ASSIGN,-----=12 (output tape-if none, leave card out)
*ASCENT,S=ULIB,N=EDITOR
*COZY
*RUN
...
(EDITOR control cards, mod packs, source decks . . .)
...
$ENDOFRUN (required last card of input deck)
*END or control cards for next job step, if any.
```

USAGE
(continued)

The limit card will usually be required only to increase the printer limit, as the default time limits will suffice for most EDITOR runs.

The assignment of unit 11 may be omitted if no files are to be retrieved from tape. The assignment of unit 12 may be omitted if no files are to be written on tape.

The file EDITOR contains a tiny ASCENT program whose only task is to retrieve a core-image, load-and-go version of EDITOR from elsewhere on ULIB, thus saving the user the time which would otherwise be spent in compilation and assembly.

When EDITOR has successfully performed the user-requested tasks specified by the control cards in the data deck and has read the "\$ENDOFRUN" control card, it executes a "CALL SYSRCL" so that other job steps may be executed.

**The Basic EDITOR
Control Cards**

All EDITOR control cards have a "\$" in column 1; the "\$" and the following characters (up to, but not including, the first blank) comprise a "command", specifying what EDITOR is to do. Required parameters, if any, are punched, free-field, in the rest of the card, separated by one or more blanks and/or by a comma. The basic control cards follow:

$\$s/d$ (where $s, d = \text{'CARD', 'DISK', or TAPE'}$)

establishes the current source s (card reader, PLIB, tape unit 11) and destination d (card punch, PLIB, tape unit 12). The command "\$DISK/DISK" is not implemented and causes an error exit.

\$MOD/FILE *name*

causes the following cards (an insert-replace-delete type of mod pack for the file *name*) to be "remembered" on a scratch unit; EDITOR can remember mod packs for 512 different files, but remembers only the latest mod pack for each.

\$PRINTING ON

sets the list-output-files-flag *loff*=1.

\$PRINTING OFF

sets *loff*=0 (assumed initially).

\$SEQUENCE *k* (where $0 < k < 10000000$)

sets the sequence-output-files-flag *soff*=*k*.

\$SEQUENCE BLANK

sets *soff*=9999999.

\$SEQCHECK *k* (where $0 < k < 10000000$)

sets the check-input-sequencing-flag *cisf*=*k*.

\$COPYFILE *namein*, *nameout*

causes the file *namein* to be read from source *s* (if *s*='CARD' and *cisf*≠0, cc 73-80 are sequence-checked). If a mod pack has been entered for *namein*, it is recalled and applied. If *soff*≠0, cc 73-80 are resequenced. If *loff*≠0, the file is listed. The file is written to destination *d* under the name *nameout*.

Note: If *nameout* is omitted, EDITOR assumes *nameout*=*namein*. Thus, "\$COPYFILE *name*" is a shorter way of saying "COPYFILE *name*,*name*".

**The Basic EDITOR
Control Cards**
(continued)

\$PROJECTNO *p*

changes the project number (*p*) under which PLIB files are read.

\$REMARK ...

is listed and otherwise ignored.

\$COMMENT ...

is listed and otherwise ignored.

\$ENDOFRUN

is the required last control card; terminates processing, properly flushing all output buffers. EDITOR then executes a "CALL SYSRCL" to recall the system for the next job step.

Basic Examples

Here are two examples of EDITOR runs using only the basic control cards:

1. Suppose that I have two large programs ("P1" and "P2") running from PLIB and that I wish to put them on tape A0001. I assign A0001 to unit 12 and use the control cards

\$DISK/TAPE	<i>s</i> ='DISK', <i>d</i> ='TAPE'
\$COPYFILE P1	copy first program
\$COPYFILE P2	copy second program
\$ENDOFRUN	done

If my PLIB files are lost, I can now regenerate them by re-assigning A0001 to unit 11 and changing the "\$DISK/TAPE" to a "\$TAPE/DISK".

2. Suppose now that I use mod packs to develop an improved version of P1 and two distinct versions of P2. I wish to put the three decks on a new tape A0002 under the names P1, P2, and P3. I assign A0001 to unit 11 and A0002 to unit 12, then use the control cards:

\$MOD/FILE P1	mod pack for P1--EDITOR
. . . .	"remembers" it on PSCR
\$MOD/FILE P2	first mod pack for P2--EDITOR
. . . .	"remembers" it on PSCR
\$TAPE/TAPE	<i>s</i> ='TAPE', <i>d</i> ='TAPE'
\$PRINTING ON	output files will be listed
\$COPYFILE P1	P1 read, modified, listed, copied
\$COPYFILE P2	P2 read, modified, listed, copied
\$MOD/FILE P2	second mod pack for P2--EDITOR
. . . .	"remembers" it on PSCR
\$COPYFILE P2,P3	P2 read, modified, listed, then
. . . .	copied under the name P3
\$ENDOFRUN	

The \$GET Control Card At Rich Helgason's request, the "\$GET" control card was implemented; it is used as follows:

One must have *s*='CARD' (input assumed from the card reader); the destination *d* may be 'CARD', 'DISK', or 'TAPE'. One defines an "alternate source" *a* (*a*='TAPE' or 'DISK', except that if *d*='DISK', then *a*≠'DISK') and an "alternate input file" *namein*, from *a*, using the control card

```
$COPYFILE namein,nameout,a
```

which establishes *a* as the alternate input source, *namein* as the alternate input file, and *nameout* as the name of the output file being constructed.

The \$GET Control Card Now, to extract cards from *namein* and add them to *nameout*,
(continued) use

\$GET k_1

to get card number k_1 from *namein*,

\$GET k_1, k_2

to get cards numbered k_1 through k_2 from *namein*,

\$GET $k_1, , namein$

to establish a new *namein* and get card k_1 from it,

\$GET $k_1, k_2, namein$

to establish a new *namein* and get cards k_1-k_2 from it.

Note: If n is the number of cards in the current *namein*, $1 \leq k_1 \leq n$ and $k_1 \leq k_2 \leq n$.

Example--\$GET

As an example, suppose that file "P1", on tape A0001, contains code to generate continental outlines on the dd80. I wish to extract this code and use it to make up a subroutine BKGRND, which I will then install in the file "P2", also from A0001, to make up a new file "P3", to be put on PLIB. At the same time, I wish to copy all three files to tape A0002. The run deck to accomplish this follows.

*JOB, ...	job card
*ASSIGN,A0001=11,R	input tape
*ASSIGN,A0002=12	output tape
.	card to get EDITOR
*RUN	control cards follow
\$CARD/DISK	s='CARD', d='DISK'
\$COPYFILE P2,P3,TAPE	a='TAPE', file names given
\$GET 1,261	get first part of P2
CALL BKGRND (P,D)	necessary subroutine call
\$GET 262,543	get the rest of P2
SUBROUTINE BKGRND (P,D)	add subroutine card
DIMENSION P(1),D(1),A(16670)	add dimensioning info
\$GET 432,512,P1	add code from P1
\$TAPE/TAPE	s='TAPE', d='TAPE'
\$COPYFILE P1	copy P1--A0001 to A0002
\$COPYFILE P2	copy P2--A0001 to A0002
\$DISK/TAPE	s='DISK', d='TAPE'
\$COPYFILE P3	copy P3--PLIB to A0002
\$ENDOFRUN	done
*END	

Binary Data Files

It is possible to generate binary data files on PLIB using the PLIB read/write routines. These files may be transferred DISK/TAPE, TAPE/TAPE, or TAPE/DISK; to do this, one uses a special form of the "\$COPYFILE", as follows:

```
$COPYFILE namein,nameout,BINARY,r
$COPYFILE namein,,BINARY,r
```

In the latter case, EDITOR assumes *nameout*=*namein*.

The integer *r* is a count of the number of records in the file. Each record must be 4096 words or less.

During such operations, the listing and resequencing flags are ignored; if either *s*='CARD' or *d*='CARD', an error message is logged and the run terminated.

**Format of an
EDITOR-produced tape**

An EDITOR-produced tape may contain up to 512 files. An EOF tape mark follows the last file on the tape. Each file consists of one or more records. The first word of the first record of a file contains the file name; the first word of each remaining record is zero. The data in each record are in words 2 through "N". When a file is copied from tape to PLIB, the first word of each record is removed; when a file is copied from PLIB to tape, the first words are added.

Each record of a normal file (except, possibly, the last) contains 512 words of data. Assume the file has r records and that the last record is n_l words long; then the data are simply a string of $5120*(r-1)+10*(n_l-1)$ characters, representing the cards of the file in COSY (COmpressed-SYmbolic) form.

Each record of a binary data file contains 4096 (or fewer) data words. The format is that assumed by the user when he wrote the file on PLIB.

**Sequencing of
Output Files**

As long as the sequence-output-files-flag *soff* is non-zero, EDITOR generates sequence numbers in cc 73-80 of each output file. Assuming $soff=k$, $k \neq 0$ and $k \neq 99999999$, the sequence numbers of cards 1, 2, 3, ... of a given output file are $0, k, 2k, 3k, \dots$ (modulo 10^8). Let n ($1 \leq n \leq 8$) be the number of digits of a particular sequence number; EDITOR packs the n digits into cc $80-n+1$ through 80 of the appropriate card, filling cc 73 through $80-n$ with the first $8-n$ characters of the output file name. Thus, with $k=10$, the 22nd card of output file 'GCMDECK' has sequence number 210; 'GCMDE210' appears in cc 73-80.

If *soff=k=9999999* (it gets that way when "\$SEQUENCE BLANK" is used), cc 73-80 of each output file are blanked.

Sequence-Checking of Input Files

If *s='CARD'* (input from the card reader), sequencing-checking may be performed. EDITOR forms a sequence number for each card from the numeric characters in cc 73-80, ignoring the non-numeric characters. As each card is read, the difference between its sequence number and the previous one is computed; if the difference is not equal to the (non-zero) value *k* last entered in the check-input-sequencing-flag *cisf*, an error comment is logged.

EDITOR Print Output

All control cards are logged. Insert, replace, and delete cards are logged following the "\$COPYFILE" card which caused the mod pack containing them to be used; when all the mods have been processed, that fact is logged, so that one can tell if the entire pack was used. Parity errors, read-length errors--in fact, detectable errors of any sort--cause an immediate bomb-out with an appropriate comment.

If listing is requested, the following comments apply: The listing of each output file begins at the top of a page with a header giving the name of the file and the date. The card images are printed in columns 41-120; each card is preceded by the number of the card in the output file. Insert, replace, and delete cards (or "\$GET" cards) are listed at their appropriate positions and start in column 2, so as to be easily seen.

Miscellaneous Notes

If a mod pack has been entered for file *name*, and *name* is used as an alternate input file (see discussion of "\$GET"), the mod pack is not applied during the "\$GET" process.

Mod packs must be sorted by the user in increasing-card-number order.

The user may wonder why "\$DISK/DISK" will not work, and why EDITOR does not allow the adding of files to an existing tape, but instead requires that a new tape be written. Both of these rules tend to protect the user from losing valuable files.

Only punched-card files should be sequenced; PLIB files and tape files should be unsequenced, as sequencing may double the length of a COSYed file.

SPECIAL CONDITIONS

Parity errors, read-length errors--in fact, detectable errors of any sort--cause execution to be terminated ("CALL EXIT"); an error comment is logged explaining the error.

I/O

In addition to the standard system units (print, punch, input, etc.), EDITOR uses the tape units 11 and/or 12, if assigned, PLIB, if PLIB files are referenced, and the system scratch unit 6762₈ (PSCR, for which no "**ASSIGN" card is required).

SPECIALIST

D. Kennison, NCAR, Boulder, Colorado 80303

LANGUAGES

FORTRAN,ASCENT

HISTORY

Written by D. Kennison in January 1971 to enable GCM to use the then-brand-new PLIB efficiently.

PROCEDURE

EDITOR has its own COSYing and deCOSYing routines which were written in ASCENT for increased speed. Files being copied are deCOSYed only if 1) they are to be listed; 2) they are to be resequenced; 3) they are to be modified; 4) they are to be punched. Every attempt has been made to overlap CPU time and PPU time. In order to minimize access time to unit 11 tape files, an in-core map of the input tape is formed, allowing most efficient access to a file previously skipped (where a choice between rewinding or backspacing must be made).

SPACE REQUIREDAbout 60000₈**TIMING**

It is difficult to provide a simple formula from which running times may be estimated. As an example, to copy tape/disk 75 GCM files totaling 341,762 COSYed words, listing all of them on microfilm (1891 frames) took about 50 CPU seconds and 58 PPU seconds on the 7600. Most tasks will be much less time-consuming than this.

PORTABILITY

Not easily portable, since it performs functions specifically tailored to the NCAR operating system.

**REQUIRED RESIDENT
ROUTINES**

PLIBRD, PLIBWT, PLIBCK, PLIBREW, PLIBNUM, SYSRCL, CLOCKF,
DATEF

PROGRAM FIDEL**LATEST REVISION** August 1972**PURPOSE**

FIDEL is a compiler for the language PDELAN. This language is an extension to FORTRAN intended for programs which use finite difference approximations for partial differential equations. It also contains commands to simplify the generation of contour plots and graphs on the microfilm plotter (dd80). It contains a macro capability, although this is not as well developed as that within the FRED preprocessor. FIDEL accepts a PDELAN source deck as input and generates a FORTRAN object program which must then be compiled by the FORTRAN compiler and executed.

The control cards required to cause FIDEL to compile a PDELAN program are the following

```

*JOB,...
  :
  *LIMIT cards for time, pages, etc.
  :
  *ASSIGN cards
  :
*ASCENT,S=ULIB,N=FIDEL
*COSY
*RUN
  :
  PDELAN program to be compiled by FIDEL
  :
*FORTRAN,S=PSCR
*COSY
*FORTRAN,S=ULIB,N=FPLOTEX,NL
*COSY
*ASCENT,S=ULIB,N=APLOTEX
*COSY
*RUN
  :
  data (if any) for PDELAN program
  :
*END

```

} Required only if
PDELAN graphics
commands are
used.

If any graphics commands are used in a subroutine, the following COMMON card must be included in that subroutine for communication with the graphics routines FPLOTEX and APLOTEX.

```

COMMON/CMDD80/NNDD80, IPDD80, HHDD80(500),
TBDD80(35,7), IZDD80

```

Additional control cards may appear between the last *COSY and the *RUN cards. Consult the PDELAN manual for more information.

PDELAN

We will provide a brief review of PDELAN. For more information the reader can consult the following NCAR documents available from the Computing Facility:

PDELAN Users Manual

Graphics Commands for the PDELAN Preprocessor: Users Manual.

DIFFERENCE OPERATORS

The language is oriented toward the solution of finite difference schemes. The user can declare a rectangular mesh and also variables on this mesh. These variables are simply arrays in FORTRAN whose dimensions are set equal to the mesh dimensions. Finite difference operators on these mesh variables can be declared. For example, we might have

```
MESH MA(100)
VARIABLE (U1,U2) ON MA
OPERATOR (MA) TO (MA(2),(100)) IS DX(W,I) =
X(W(I+1)-W(I-1))/DLX
```

These are all declarations. The operators can only be used within the range of a DOMESH. The DOMESH causes the statements within the range of the DOMESH to be executed over the portion of the mesh indicated in the DOMESH statement. Use of the DOMESH permits a subscript free notation to be used. For example

```
DOMESH 30 MA(K=(2,100))
30 U2=U1-DLT*U1*DX(U1)
```

FIDEL will translate this into the following FORTRAN code (actually the output will be slightly different than shown below)

```
DO 30 K=2,100
30 U2(K)=U1(K)-DLT*U1(K)*(U1(K)-U1(K-1))/DLX
```

GRAPHICS COMMANDS

The language contains high level graphics commands which permit contour plots or graphs to be generated. Labels can be easily placed on the plots and several plots can be placed on a single frame either separately or overlaid. The options such as grid background, divisions, tick marks, line patterns, etc. are set by default. These defaults can be overridden by additions to the GRAPH command. For example, given the two dimensional array U, a contour plot can be generated by

```
GRAPH (U)
```

If U is 2 dimensional

```
REAL U(40,40)
```

then a portion of the array will be plotted by the command

```
GRAPH (U(1..20,21..40))
```

Two plots can be placed on separate frames by the command

```
GRAPH (U/V)
```

A graph of the variables

```
REAL X(100), Y(100), W(100,5)
```

can be obtained by

```
GRAPH (Y/(X,Y))
```

or

```
GRAPH (Y(1..N)/W(,3)/W(1..N,K))
```

In the first command the points Y(I) for I = 1 to 100 are graphed on the first frame and (X(I),Y(I)) for I = 1 to 100

are graphed on the second frame. A heading can be placed on the graph by a command of the following form

```
GRAPH (U) HEAD = (FLIST = (*U AT STEP*,M))
```

The character string "U AT STEP" will be printed along with the name and value of the variable M. The printing will be centered within the plot area. This provides a format-free printing capability for plot generation.

More than one plot may be placed on the same frame. For example, the following command will place a contour plot of U on the top half of the frame and a contour plot of V on the bottom half.

```
REAL U(40,40), V(50,50)
  ⋮
GRAPH (U,V) PLOT = (PIC = TOP), (PIC = BOT)
```

There are many possible variations and options within the GRAPH command. We refer the reader to the users manual for a more complete description.

SPECIALIST

Russell Rew, NCAR, Boulder, Colorado and John Gary, Computer Science Department, University of Colorado, Boulder, Colorado 80303

LANGUAGE

PDELAN, FORTRAN, ASCENT

HISTORY

FIDEL was designed and implemented by Richard Helgason, Gyula Lócs and John Gary during 1971 and 1972.

TIMING

This is highly variable. For some programs, FIDEL has compiled at the rate of 500 cards/minute.

PORTABILITY

FIDEL is highly nonportable. It is completely tied to the CDC 6600 - 7000 series and would be somewhat difficult to move even from the NCAR system to the CDC system.

SUBROUTINE FLEX (IIN,IOT)

LATEST REVISION March 1974

PURPOSE File management of COSYed PLIB card files. FLEX merges cards from the card reader (unit 5) with cards from a specified input file to form an output file which may be written to PLIB or PSCR, written to a logical unit, or punched.

ACCESS CARDS *FORTRAN,S=ULIB,N=FLEX
 *COZY

USAGE Call FLEX (IIN,IOT)

ARGUMENTS IIN
 Specifies the input file.
 = 1,...,4,8,...,4095 - input from specified unit
 = *nHexxxx...* - input from specified PLIB file
 = 0 - no input file

ARGUMENTS
(continued)

IOT

Specifies the output file.

- = 1,...,4,8,...,4095 - output to specified unit
- = 7 - output punched
- = *nHxxxx...* - output to specified PLIB files.
- = 4HPSCR - output to "PSCR" (6762₈)
- = 0 - no output unit

The contents of the arguments IIN and IOT are unchanged by FLEX.

Card Reader Input

FLEX reads cards from the card reader (unit 5); each card thus read is sent to the output file, unless it has a "\$" in column 1, in which case it is assumed to be a control card, directing FLEX to perform a particular task.

FLEX Control Cards

A FLEX control card has a "\$" in column 1 and one of fourteen possible three-character command words in columns 2-4. The fourteen possible commands are as follows:

\$GET

Used to get cards from the input file and write them to the output file.

\$SET

Used to define a following deck of up to 64 cards as a named "set".

\$EST

Used to mark the end of a "set" when it is inconvenient to use one of the other control cards.

\$USE

Used to retrieve a specified "set" of cards and send them to the output file.

\$DEL

Used to define a DPC string to be used in deleting cards from the input stream or the input file.

\$EDL

Used to end those deletions from the input stream or the input file due to a particular DPC string.

\$SEQ

Used to activate sequencing (in cc 73-80) of cards sent to the output file.

\$SYM

Used to reset certain sequencing parameters.

\$NUM

Used to reset certain sequencing parameters.

\$INC

Used to reset certain sequencing parameters.

\$ESQ

Used to suspend sequencing of the output file.

\$SUP

Used to suppress listing of all but control cards. (Normally, all output cards are listed.)

\$ESP

Used to re-activate normal listing.

\$END

Last card read by FLEX--causes output file to be terminated properly and FLEX to execute a "RETURN".

FLEX Control Cards (continued)

Many of the above control cards require "parameters", which are punched, free-form, in columns 5-80 of the control card. Parameters must be separated by any combination of commas and blanks, and are of two types:

- a "name", punched as a string of from 1 to 8 alphanumeric characters, with no imbedded blanks.
- a "number", punched as a string of from 1 to 8 numeric characters, with no imbedded blanks.

The \$GET Card

\$GET n_1, n_2

\$GET n_1

\$GET

cause FLEX to get cards from the input file and send them to the output file. If n_1 and n_2 are specified, cards numbered n_1 through n_2 will be copied. If only n_1 is specified, only the card numbered n_1 will be copied. If neither n_1 nor n_2 is specified, then all remaining cards in the input file will be copied; in particular, if "\$GET" and "\$END" are the only two control cards read by FLEX, then the entire input file is copied to the output unit.

Note: The numbers n_1 and n_2 reference the ordinal numbers of cards in the input file (from 1 to N in increments of 1); they have nothing to do with sequence numbers, if any, in cc 73-80.

Note: If $n_1 > N$, the \$GET does nothing but space through the input file until the end-of-file is reached; no cards are sent to the output file. If $n_1 \leq N$, but $n_2 > N$, cards numbered n_1 through N are copied to the output file.

The \$SET Card*\$SET name*

causes FLEX to create a temporary file called *name* and to store in this file all the cards which follow the \$SET card up to, but not including, the next FLEX control card. (\$EST may be used to mark the end of the set if it is inconvenient to end it on one of the other control cards.) No set may contain more than 64 cards; no more than 32 sets may be defined during the course of a single FLEX call.

The \$USE Card*\$USE name*

causes FLEX to retrieve the set of cards identified by *name* and send them to the output file.

The \$DEL Card*\$DEL code*

causes the DPC string *code*, *m* characters long ($1 \leq m \leq 8$) to be entered in a 32-position "deletion table". Subsequently, every card from the input stream or from the input file is examined to see if card columns 73 through $73+m-1$ contain the character string *code*; if so, the card is deleted. Up to 32 delete codes may be activated at any one time.

The \$EDL Card*\$EDL code*

causes the DPC string *code* to be removed from the "deletion table" (in which it was previously entered by means of a "\$DEL" control card).

The Sequencing Control Cards

The output file may be sequenced in cc 73-80. When activated, sequencing is performed under control of the parameters *sym*, *num*, and *inc*, where *sym* is a DPC string of eight or fewer characters and *num* and *inc* are integers. The following steps are performed for each output card:

1. The DPC string *sym* is left-justified in cc 73-80;
2. $num=num+inc$;
3. *num* is converted to DPC, with leading zeroes, and stuffed into the remaining card columns, following *sym*. If *num* gets too big to fit, its high-order digits are simply lost.

Initially, FLEX assumes sequencing deactivated, *sym* null, *num*=0, and *inc*=1. If sequencing is activated using these default values, the sequence numbers generated will be "00000001", "00000002", etc.

The control cards having to do with sequencing may seem somewhat overdone; Helgason set them up that way to allow each user to suit his own tastes.

The \$SEQ Card

```
$SEQ sym,num,inc
$SEQ sym,num
$SEQ sym
$SEQ
```

are used to activate sequencing and reset selected parameters.

The \$SYM Card\$SYM *sym,num,inc*\$SYM *sym,num*\$SYM *sym*

\$SYM

do not affect the activated/deactivated state of sequencing; they only reset selected parameters. The last, with no specified parameters, has the special effect of setting *sym* to eight DPC blanks, thus causing cc 73-80 of subsequent output cards to be blanked.

The \$NUM Card\$NUM *num,inc*\$NUM *num*

do not affect the activated/deactivated state of sequencing; they are used only to reset selected parameters.

The \$INC Card\$INC *inc*

is used to reset the sequencing parameter *inc*; it does not affect the activated/deactivated state of the sequencing.

The \$ESQ Card

\$ESQ

is used to deactivate sequencing. The parameters *sym*, *num*, and *inc* are unchanged. If sequencing is subsequently reactivated, it resumes from where it left off.

The \$SUP Card

\$SUP

is used to suppress listing of the output file. Control cards are still listed.

The \$ESP Card

\$ESP

is used to resume listing of the output file.

The \$END Card

\$END

causes FLEX to properly terminate writing of the output file and to execute a "RETURN."

FLEX Initialization

Note that, each time FLEX is called, it initializes itself completely. Card sets defined by "\$SET" control cards and delete codes defined by "\$DEL" control cards during a previous call are lost; listing is turned on, sequencing off.

ENTRY POINTS

FLEX, PARM0, PNUM, PNAM, PUT, OUT, PULL, ENCOSY, FLUSHB, FLUSHC, FLUSHE, FLUSHS, DECOSY, EMPTY, and FULL

SPECIAL CONDITIONS

Almost any sort of detectable error (control card errors, parity errors, read-length errors, etc.) will cause FLEX to print an error comment and execute a "CALL EXIT".

COMMON BLOCKS

UNITS (4₈), SETS (1101₈), CARD (10₈), DELS (101₈), SEQN (5₈), FILE (2₈), OUTS (1001₈), GETS (4₈), PARM (16₈), ENCOS (132₈), and OVL P (1000₈)

I/O

It may be of interest to the user to know in what form his files are stored on the various I/O devices:

**Card Reader
and Punch**

Cards are read from the card reader into an eight-word buffer (or punched from an eight-word buffer) under control of an "8A10" format, giving 10 six-bit DPC characters per word.

Numbered Unit

A file read from or written to a numbered unit (which may be assigned to tape, to the disk, or, on the 6600, to a drum) consists of a string of 512-word records, each containing 64 cards in "8A10" format; an EOF mark follows the records. (Except that, if the number of cards in the file (n) is not a multiple of 64, then the last record contains m cards and is $8m$ words long, where $m=n$ modulo 64). These records are read/written in odd parity, using "BUFFER IN/BUFFER OUT" statements.

**PLIB and PSCR
(COSYed files)**

A file on PLIB or PSCR is in COSY (COmpressed SYmbolic) form. It consists of a string of n records, the first $n-1$ of which are 512 words long and the last of which is m words long, where $1 \leq m \leq 512$. These records contain $(5120(n-1)+10m)$ six-bit COSY characters (packed 10 per word), representing the cards of the file. These COSY characters are defined as follows:

00 ₈	end-of-card code
01 ₈ through 32 ₈	DPC 01 ₈ through 32 ₈ --A through Z
33 ₈ through 44 ₈	DPC 33 ₈ through 44 ₈ --0 through 9
45 ₈ through 63 ₈	DPC 45 ₈ through 63 ₈ --+,*,/, etc.
64 ₈	two blanks
65 ₈	three blanks
⋮	⋮
74 ₈	ten blanks
75 ₈	twenty blanks
76 ₈	thirty blanks
77 xx ₈	the DPC character xx ₈ , where xx ₈ =00 ₈ , 64 ₈ , 65 ₈ ,...77 ₈
770100 ₈	COSY end-of-file

**PLIB and PSCR
(COSYed files)**
(continued)

Note that a COSYed file is, in general, much shorter than the unCOSYed file it represents, for two reasons:

1. Strings of blanks are tremendously compressed.
2. Blank strings ending in column 80 are removed entirely. For example, note that the two 80-column cards

```
cc      7                               73      80
      A=B+C      . . .      DECK0002
      D=E+F      . . .
```

may be represented by the 25 COSY characters

70 ₈ - 6 blanks	}	COSY word 1	}	1st Card
01 ₈ - A				
54 ₈ - =				
02 ₈ - B				
45 ₈ - +				
03 ₈ - C				
76 ₈ - 30 blanks				
76 ₈ - 30 blanks				
55 ₈ - 1 blank				
04 ₈ - D				
05 ₈ - E	}	COSY word 2	}	2nd Card
03 ₈ - C				
13 ₈ - K				
33 ₈ - 0				
33 ₈ - 0				
33 ₈ - 0				
35 ₈ - 2				
00 ₈ - end-of-card				
70 ₈ - 6 blanks				
04 ₈ - D				
54 ₈ - =	}	partial COSY word 3	}	2nd Card
05 ₈ - E				
45 ₈ - +				
06 ₈ - F				
00 ₈ - end-of-card				

The above example illustrates another important point-- that sequencing the cards of a file in cc 73-80 may increase the COSYed length of the file significantly.

FLEX has its own built-in enCOSYing and deCOSYing routines, written in FORTRAN, to process COSYed files.

Printer Output

Control cards are always listed by FLEX; cards sent to the output file are also listed, unless the \$SUP card is used to suppress that part of the listing.

**REQUIRED ULIB
ROUTINES**

None

SPECIALIST

D. Kennison, NCAR, Boulder, Colorado 80303

LANGUAGE

FORTRAN

HISTORY

Written by R. Helgason about April or May, 1971.
Commented and worked over by D. Kennison in March 1974.

SPACE REQUIRED

about 12000₈

TIMING

It is almost impossible to write a formula describing the timing of FLEX; a few seconds of CPU/PPU time will suffice for most tasks. The only way to speed it up would be to program many of its functions into ASCENT.

PORTABILITY

FLEX was specifically tailored to the NCAR system and is largely unportable except as an idea.

**REQUIRED RESIDENT
ROUTINES**

RANINT, RANRD, RANWT
SBYTE, GBYTE, SBYTES, GBYTES
PLIBREW, PLIBRD, PLIBWT, PLIBCK

PROGRAM FRED**LATEST REVISION** October 1, 1973**PURPOSE**

The program FRED is a precompiler; given an input file of routines written in the FRED "language" (FORTRAN with certain extensions), it produces an output file, in FORTRAN, suitable for input to the compiler. It provides the user conveniences such as conditional compilation, macros, debugging aids, evaluation of linear subscript expressions and renumbering of statement labels. Users are strongly urged to recommend additions to FRED's repertoire.

USAGE

A FRED run deck has the following format:

```
*JOB,...
  .
  .
  .
  limits and assignments for your execution
  .
  .
  .
*ASCENT,S=ULIB,N=FRED
*COSY
*RUN
  .
  .
  .
  FRED input file
  .
  .
  .
*FORTRAN,S=PSCR{,options}
*COSY
  .
  .
  .
  other compilations and/or assemblies for your execution
  .
  .
  .
*RUN
  .
  .
  .
  data for your execution
  .
  .
  .
*END
```

The "***FORTRAN,S=PSCR**" card may have options as desired:
 "FL" to list the FRED output, "PS" to punch a source deck,
 "PC" to punch a COSY deck, "PB" to punch a binary deck, etc.

Input

The input file consists of all cards following the "***RUN**" which caused execution of FRED, including cards retrieved from ULIB or PLIB by "***MOD**" cards, if any, up to and including a ".FINISH" card. It is assumed to contain one or more routines, each beginning with a "header" card ("PROGRAM", "SUBROUTINE", etc.) and terminated by an "END" card. Control cards (other than ".FINISH", of course) may be inserted anywhere in the input file except between the first card of a statement and one of its continuation cards; they

are used to control input file listing, to define macros, to effect conditional compilation, to request renumbering, etc.

Control Card Format

All FRED control cards have a period in column 1, followed by a FRED control word (starting in column 2, with no imbedded blanks), followed by one or more blanks, followed by parameters, if any. Columns 73-80 are not read; do not use them. If more than 72 characters are required, a comma in column 1 of the next card allows for continuation.

The control cards currently implemented are as follows (the { } symbols are used to delimit optional parameters):

.FINISH	
.FINISH	GO
.ACTIVATE	<i>alphabetic character list</i>
.DEACTIVATE	<i>alphabetic character list</i>
.ELIMINATE	<i>alphabetic character list</i>
.IF	<i>expression</i>
.OTHERWISE	
.ENDIF	
.MACRONAME	<i>name</i> { (a_1, \dots, a_n) } = <i>expression</i>
.GLOBAL	<i>name</i> { (a_1, \dots, a_n) } { l_1, \dots, l_m }
.INSERT	<i>name</i> { (e_1, \dots, e_n) } { s_1, \dots, s_m }
.LOCAL	{OFF}
.RENUMBER	
.RENUMBER	n_1, n_2
.RENUMBER	OFF
.LIST	
.LIST	OFF
.TOPOFFORM	
.PLIB	<i>name</i>
.CHECK	
.CHECK	{SUBSCRIPTS} {VALUES} {FLOW}
.CHECK	OFF

Control Card Format
(continued)

Control card recognition is based on the first two characters of the control word (".MA", ".MAMA", and ".MARVELOUS" are all equivalent to ".MACRONAME"). Thus, one may use the following short-hand forms:

.AC	.EN	.IN	.OT
.CH	.FI	.LI	.PL
.DE	.GL	.LO	.RE
.EL	.IF	.MA	.TO

The .FINISH Card

The last card of your input file must be the control card

```
.FINISH {GO}
```

The optional parameter "GO", if used, forces FRED to recall the system for another job step even if errors were detected in the input file (in which case, FRED would normally terminate the run with a "CALL EXIT").

Column 1 Control

A type of conditional compilation is provided by the following:

Input cards which have an alphabetic character in column 1 are treated in a special manner. Each of the 26 characters so used is in one of three states - activated, deactivated, or eliminated - determining how the card should be treated. If the character in column 1 is activated, it is replaced by a blank and the card is treated as a FORTRAN source card. If the character in column 1 is deactivated, it is replaced by a "C" and the card thus becomes a comment card. If the character in column 1 is eliminated, the entire card is ignored.

All 26 alphabetic characters are initially deactivated - to change the state of a set of characters, use the control cards

- .ACTIVATE *alphabetic character list*
- .DEACTIVATE *alphabetic character list*
- .ELIMINATE *alphabetic character list*

Non-alphabetic characters in the list (blanks and commas, in particular) are ignored and may be used to improve readability.

Example:

```
.ACTIVATE A,B
A      F=Y
C      F=Z
.ELIMINATE D
B      WRITE (6,100)
B 100  FORMAT (* THIS IS IT*)
      X=(F
B      1      +Y
D      2      +Z
      3      )/6.
```

gives the following result:

```
      F=Y
C      F=Z
      WRITE (6,100)
100  FORMAT (* THIS IS IT*)
      X=(F
      1      +Y
      3      )/6.
```

**Conditional
Compilation**

A conventional type of conditional compilation is provided by the ".IF", ".OTHERWISE", and ".ENDIF" control cards, used as follows:

```
.IF e      (where e is an expression to be evaluated by
           FRED)
. . .     Code to be used if e is true (e≠+0) -
           ignored if e is false
.OTHERWISE (This card may be omitted if no code for e
           false)
. . .     Code to be used if e is false (e=+0) -
           ignored if e is true
.ENDIF    (Terminates the current ".IF")
```

Both the *e*-true and *e*-false code blocks may contain control cards (except for ".FINISH", of course) as well as source cards, thus opening up a great many possibilities. In particular, ".IF" blocks may be nested eight levels deep.

The following rules define expressions legal for use on ".IF" control cards:

1. Decimal integers (0 to $2^{59}-1$), octal constants (*xxxx...xxx*B), and logical constants (.TRUE.≡-0 and .FALSE.≡+0) are legal expressions.
2. A previously-defined macro is a legal expression provided that its defining expression is legal.
3. A quoted expression is legal, provided that its evaluation results in a single decimal integer (signed or unsigned).
4. If *e* is a legal expression, then so are (*e*), .NOT.*e*, +*e*, and -*e*, except as prohibited by rule 6.
5. If *e* and *f* are legal expressions, then so are *e***f*, *e*/*f*, *e*+*f*, *e*-*f*, *e*.EQ.*f*, *e*.NE.*f*, *e*.LT.*f*, *e*.LE.*f*, *e*.GT.*f*, *e*.AND.*f*, and *e*.OR.*f*, except as prohibited by rule 6.
6. If two operators are consecutive, the second must have priority equal to or greater than that of the first. ("-.NOT." is illegal, for example.)

Expressions are evaluated as follows: First, each macro not occurring in a quoted expression is replaced by its defining expression - this step repeats until no such macros remain. Then, each quoted expression is replaced by its integer value. Finally, the result (containing only parentheses, constants, and operators) is evaluated using standard parenthesis conventions and the following operator hierarchy (from high priority, evaluated first, to low priority, evaluated last):

Unary + and -	Integer no-op and 60-bit complement respectively.
* and /	Integer multiply and divide, respectively.
	* yields shift if either operand is a power of 2.
	/ yields zero when the divisor is zero.
Binary + and -	Integer sum and difference.
Relationals	(.EQ., .NE., etc.) result -0 if true, +0 if false.
.NOT.	60-bit complement
.AND.	60-bit Boolean product.
.OR.	60-bit Boolean sum.

Miscellaneous Notes:

- Operators of equal priority are evaluated from right to left.
- The operators ".NOT.", ".AND.", and ".OR." perform masking or logical operations, depending on the way in which they are used.
- The evaluator does not assume a macro to be enclosed in implied parentheses - thus, $3 * \text{SUM}(1,2)$, where $\text{SUM}(I,J) = I + J$, evaluates as $3 * 1 + 2 = 5$, rather than $3 * (1 + 2) = 9$ - unless said macro occurs as part of a quoted expression.

Macro Facilities

(*Note:* Throughout this section, assume "*name*" and "*a_k*" to be legal FORTRAN names - 7 or fewer alphanumeric characters, the first alphabetic.)

Three types of macros are provided for by FRED - first, the control card

`.MACRONAME name{(a1,a2,...,an)} = expression`

defines a macro named "*name*", with formal arguments, if any, *a₁*, *a₂*, ..., *a_n*. The macro is available for use throughout the rest of the input file. Whenever "*name*" occurs in a legal position, it is replaced by its defining expression; formal arguments occurring in that expression, if any, are replaced by real arguments (themselves expressions and given, in parentheses, following the occurrence of "*name*"). The phrase "legal position" is difficult to define; it is closely equivalent to "any position in which a name or a constant might appear".

The second type of macro is defined by the extended-FORTRAN statement (starting in column 7)

`MACRONAME name{(a1,a2,...,an)} = expression`

and is available for use only from the point of definition up to the next "END" statement in the input file - otherwise, it is identical to the first type.

Example:

```

.MACRONAME N=10
.MACRONAME SRSMSQ(X,Y)=SQRT(X**2+Y**2)
.MACRONAME SUB=COMPUTE
  MACRONAME AF(P,Q,K)=(P(K)+Q(K))
  MACRONAME SF(P,Q,K)=(P(K)-Q(K))
PROGRAM TEST
COMMON A(N),B(N),C(N)
READ (5,1) (A(I),B(I),I=1,N)
1 FORMAT (8F10.0)
DO 2 I=1,N
C(I)=SRSMSQ(AF(A,B,I),SF(A,B,I))
2 CONTINUE
CALL SUB $ CALL EXIT
END
...   ...   ...   At this point, AF and SF are no longer
                        defined.

```

gives the following result:

```

PROGRAM TEST
COMMON A(10),B(10),C(10)
READ (5,1) (A(I),B(I),I=1,10)
1 FORMAT (8F10.0)
DO 2 I=1,10
C(I)=SQRT((A(I)+B(I))**2+(A(I)-B(I))**2)
2 CONTINUE
CALL COMPUTE
CALL EXIT
END

```

The third type of macro is called a global; it involves more than one card and is defined by

```

.GLOBAL name{a1,a2,...,an} {l1,l2,...,lm}
.      .      .
.      (FORTRAN statements)      .
.      .      .
.
(A "." in column
1 => end of global)

```

It is available for use throughout the rest of the input file.

Macro Facilities
(continued)

The control card

```
.INSERT name{(e1,e2,...,en)} {s1,s2,...,sm}
```

causes the FORTRAN statements comprising the global "name" to be inserted at that point in the code. Each occurrence of a formal parameter a_k is replaced by the defining expression e_k . Each occurrence of a dummy statement label l_k is replaced by the corresponding "real" statement label s_k . Any or all of the s_k may be omitted; in that case, the corresponding l_k are replaced by unique statement labels generated by FRED.

Note: The s_k list may thus begin with a comma, if l_1 is to be uniquely generated, and may contain consecutive commas. Characters other than ",", "0-9" may be used to improve readability - for example, "-,-,13,-,26" may be used instead of ",,13,,26". Trailing commas may be omitted.

Example:

Suppose we define the global "SUMMER" using

```
.GLOBAL SUMMER (X,N,D,S) 1,2
  S=0.
  DO 2 I=1,N
  IF (X(I)) 2,999,1
  1 S=S+X(I)/D
  2 CONTINUE
  IF (S.EQ.0.) GO TO 998
```

Then, the subsequent control card

```
.INSERT SUMMER (P,100,(A+B),SUMP)
```

gives the result

```
SUMP=0.
DO 10001 I=1,100
IF (P(I)) 10001,999,10000
10000 SUMP=SUMP+P(I)/(A+B)
10001 CONTINUE
IF (SUMP.EQ.0.) GO TO 998
```

The control card

```
.INSERT SUMMER (P,100,(A+B),SUMP) 10,20
```

would give the same result, except that 10 and 20, rather than 10000 and 10001, would replace 1 and 2.

The .LOCAL Card

To temporarily suppress recognition of all macros defined by ".MACRONAME" control cards, use the control card

```
.LOCAL
```

to begin recognition again, use the control card

```
.LOCAL OFF
```

Evaluations of Linear Expressions (Quoted Expressions)

The ' symbol is used to enclose expressions which are to be evaluated by FRED *to obtain a constant or a linear combination of one or more simple variables.* Such expressions may contain grouping parentheses, decimal or octal constants, non-subscripted variable names, macros, other quoted expressions, and any of the integer arithmetic operators +, -, * and /. This feature may be used to good advantage in finite difference subscript coding and in changing dimensions.

**Evaluations of
Linear Expressions
(Quoted Expressions)**
(continued)

Example:

```
.MACRONAME X(I)=I-1
.MACRONAME M=66
MACRONAME N='M-2'
DIMENSION R('M-1',M),S('2*M'),T(N)
R(J)=S('2*(X(J)+6)')
```

gives the following result

```
DIMENSION R(65,66),S(132),T(64)
R(J)=S(10+2*J)
```

The quoted-expression evaluator uses standard parenthesis conventions. The operators * and / are evaluated first, + and - after; equal priority operators are evaluated from left to right. Implicit parentheses are assumed around macros and imbedded quoted expressions - thus, 3*SUM(1,2), with SUM(I,J)=I+J, and 3*'1+2' both evaluate as 3*(1+2)=9, rather than 3*1+2=5. The operator * will give incorrect results if the product is greater than 2**59-1. Division by zero yields zero.

**Statement
Renumbering**

The control card

```
.RENUMBER {n1,n2}
```

activates renumbering and (optionally) provides a starting number n_1 and an increment n_2 . If n_1 and n_2 are omitted, FRED assumes $n_1=n_2=1$. The control card

```
.RENUMBER OFF
```

deactivates renumbering. Renumbering takes place for a given routine when its "END" card is processed - at that point, if renumbering is activated, it is performed using the current values of n_1 and n_2 .

Input Listing

Normally, FRED does not list the input; however, any routines with errors are listed in their entirety, along with appropriate error comments. In addition, the control card

```
.LIST
```

will cause subsequent statements to be listed, until the control card

```
.LIST OFF
```

is encountered.

To cause a page eject during listing, use the control card

```
.TOPOFFORM
```

Note: These control cards simply insert list control flags in the list file; listing of cards from a given routine actually takes place when its "END" card is processed.

Each input card listed is preceded by its number in the input file, written in the form *xxxx*.0000; cards inserted by a ".INSERT" control card are numbered *xxxx*.0001, *xxxx*.0002, etc.

Output

FRED produces no listing of its output; the output file must be passed through the compiler to obtain a listing.

Normally, the output file is written to "PSCR"; however, it is possible to write it to one or more PLIB files. The control card

```
.PLIB name
```

directs FRED to write the output for subsequent routines to the file *name*. In order to avoid problems with interleaved reading and writing of PLIB (which does not always work) the

Output
(continued)

only immediate result of the ".PLIB" control card is to flush the output file buffer to "PSCR", followed by a special "signal record", containing the PLIB name. When the ".FINISH" card is reached, the contents of "PSCR" are re-read and parcelled out to the various PLIB files.

All 80 columns of comment cards are preserved in the output file. Columns 73-80 of source cards are lost; blanks not imbedded in variable names, constants, or FORTRAN keywords are preserved. Statement-separating dollar signs are not used; each new statement begins on a new card.

Debug Facilities

FRED provides three types of user-run-time debugging: subscript-checking, value-checking, and flow-checking. It does this (when requested) by inserting calls to checking routines at appropriate points in the user's code, as follows:

Subscript Checking: A "CALL SBRGCK" is inserted prior to any executable statement in which subscripts appear (except when they appear in an I/O list). The call may have from three to seven arguments: an integer "checkpoint", which increments by one from one SBRGCK call to the next, followed by at least one, and not more than three, pairs of integers, the first number of each pair being a dimension and the second the subscript to be compared with that dimension.

Value Checking: A "CALL VARGCK" is inserted following every arithmetic replacement statement. The call may have from three to six arguments: the Hollerith name of the variable appearing on the right-hand side of the statement, followed by an integer type indicator (1 => logical, 2 => integer, 3 => real, 4 => double, and 5 => complex), followed by the value of the variable, followed by up to three subscript values, if any.

Flow Checking: A "CALL FLRGCK" is inserted in place of any executable statement having a statement label; the executable statement is moved down to follow the call. The call has exactly two arguments: the integer value of the statement label, and the Hollerith name of the routine in which it appears.

Example:

Assuming all three types of checking:

```
PROGRAM TEST
  DIMENSION A(2,3,4),M(2,4,3)
  . . . . .
12 M(I,K,J)=A(I,J,K)/10.
```

Statement 12 translates into the following (assume 43 previous calls to SBRGCK):

```
12 CALL FLRGCK (12,4HTEST)
   CALL SBRGCK (44,2,I,4,K,3,J)
   CALL SBRGCK (45,2,I,3,J,4,K)
   M(I,K,J)=A(I,J,K)/10.
   CALL VARGCK (1HM,2,M(I,K,J),I,K,J)
```

The system library contains standard versions of SBRGCK, VARGCK, and FLRGCK. The system SBRGCK prints a warning only if a subscript is out of range. The system VARGCK prints one line (examples: "M(1,3,2)=4", "A=46."). The system FLRGCK prints one line (example: "FLOW - 16 IN MATRIX"). It may sometimes be to the user's advantage to provide his own versions of these routines (for example, a run could be terminated if any entry in an array exceeded a certain value).

The insertion of calls to SBRGCK, VARGCK, and FLRGCK is a two-pass process:

- Pass 1 is performed for a given routine if, and only if, checking was in an activated state when the header card for that routine was read from the input file. It

Debug Facilities
(continued)

involves modifications of DO-loops and FORTRAN IV statements of the form "IF (r) s" - these modifications are introduced as the routine is read and appear in the intermediate output file for the routine, together with control flags for pass 2.

Pass 2, the actual insertion of the calls, takes place when an "END" card is encountered for a routine for which pass 1 was performed - the calls are inserted in the intermediate output file as specified by control flags inserted during pass 1.

The control card

```
.CHECK {SUBSCRIPTS} {VALUES} {FLOW}
```

activates checking and, if pass 1 is being performed for the routine in which it appears, places a pass 2 control flag in the intermediate output file - the control flag activates the types of checking specified by the parameter list and deactivates the rest.

The control card

```
.CHECK OFF
```

deactivates checking - it should be used between the "END" card of a routine for which pass 1 was performed and the header card of the next routine, for which pass 1 is not to be performed.

Note: Parameters must be separated by at least one non-alphabetic character - also, since parameters are recognized by their first character, the parameters "S", "SAMSON", and "SACERDOTAL" are all equivalent to "SUBSCRIPTS".

The above discussion may seem rather formidable, but the control cards are very easy to use. For example, to get subscript-checking throughout an entire file, just insert the card ".CHECK SUBSCRIPTS" (or ".CH S") before the header card of the first routine. To value-check and flow-check a section of a routine, insert ".CHECK" (or ".CH") before the header card of the routine, ".CHECK VALUES, FLOW" (or ".CH V, F") before the section to be checked, ".CHECK" (or ".CH") after the section, and ".CHECK OFF" (or ".CH O") after the "END" card of the routine.

SPECIAL CONDITIONS

FRED may (rarely) stop after printing one of the following diagnostics:

ERROR IN COMPUTED GO TO
 EDITOR LOGIC ERROR - DUMP (followed by a core dump)
 TABLE OVERFLOW - DUMP (followed by a core dump)
 BUFFER ERROR ON *xxxxxxx* - STOP
 STOP PREPROCESSOR
 COSY OUTPUT FAILURE

Do the following:

- Examine the two or three cards preceding the "last card read" for misspunches or damage.
- Make sure the input file is terminated by a ".FINISH".
- Run the deck again if the problem was a buffer error.
- See the FRED consultant.

I/O

FRED uses the following system units (all of which are automatically assigned):

- 6761₈ - diagnostic file
- 6762₈ - output file (PSCR)
- 6763₈ - list file
- 6764₈ - intermediate file
- 6776₈ - printer output (FORTRAN logical unit 6)
- 6777₈ - card reader input (FORTRAN logical unit 5)
- 7001₈ - PLIB (when ".PLIB" control cards are used)

SPECIALIST

David Kennison, NCAR, Boulder, Colorado 80303

LANGUAGES

FRED, FORTRAN, and ASCENT

HISTORY

Written at NCAR by Richard V. Helgason during 1972 as an off-shoot of FIDEL (FInite Difference Equation Language). Taken over by D. Kennison about March 1, 1973.

PROCEDURE

The procedure used by FRED is perhaps most easily described in terms of the files it uses, as follows:

The list file contains a copy of all input file cards encountered since the last "END" card was processed, each with an associated card number and a list control flag.

Each ".INSERT" control card is followed by a copy of the cards belonging to the global referenced. When an "END" card is processed, the list file is used if, and only if:

- Errors were found in the routine. (The diagnostic file is non-empty.)
- The user requested an input listing for any part of the routine.

The diagnostic file receives error messages for the current routine, each with a card number linking it to a card in the list file and a column number giving the approximate location of the character being examined when the error was discovered. If the diagnostic file is non-empty when an "END" card is processed, the error messages it contains are merged with the list file and the result printed.

The intermediate output file contains pointer strings representing the FORTRAN statements and the comment cards of the output form of the current routine. As a routine is read from the input file, each statement is scanned - its type is determined, macros and expressions in quotes are evaluated, statement labels and names are saved in hash-coded tables, errors are detected and logged in the diagnostic file. The final form of each statement is a string of 18-bit pointers, where a given pointer may:

- Reference a FORTRAN-keyword table;
- Reference a statement-label table;
- Reference a name table;
- Reference a single-character table;
- Represent a count of n blanks;
- Signal a Hollerith string to follow.

PROCEDURE
(continued)

Comment cards are translated into strings of pointers of types 4, 5, and 6. These pointer strings accumulate in the intermediate output file until the "END" card is reached, when they are retranslated to generate the output form of the routine. Note that by changing a single table entry just prior to retranslation, all references by the routine to a given variable name or a given statement label (or, for that matter, a given FORTRAN keyword) are changed. This fact is used to effect statement label renumbering and could be used in various other ways.

The output file accumulates the FORTRAN routines output by FRED, together with "flag records" specifying the PLIB files, if any, to which the routines are subsequently to be written.

SPACE REQUIRED

About 100000₈

TIMING

Assemble-and-load time for FRED is negligible. Execution time may vary considerably; a reasonable rule of thumb is to allow about one second CPU time and $\frac{1}{2}$ second PPU time for every hundred cards in the input file.

PORTABILITY

FRED is not easily portable; its present curator is looking into this. Major problems:

- 1302 lines of ASCENT.
- The assumption throughout of 6-bit characters, 60-bit words, NCAR's particular DPC character set, and NCAR's particular brand of FORTRAN.