

SLATEC1: Overview and Subject Guide

Table of Contents

Preface	4
Background on the SLATEC Library	5
Using the SLATEC Library	6
SLATEC Error Procedure	8
B-Splines	10
Abstract	10
Description of B-Splines	10
Basic Routines	11
Interpolation	12
Differentiation and Integration	12
Extrapolation	13
Curve Fitting and Smoothing	13
Routines in the B-Spline Package	13
References	13
EISPACK	15
Special Functions Background	16
Intrinsic Functions and Fundamental Functions	17
Elementary Functions	17
Trigonometric and Hyperbolic Functions	17
Exponential Integrals and Related Functions	18
Gamma Functions and Related Functions	18
Error Functions and Fresnel Integrals	19
Bessel Functions	19
Bessel Functions of Fractional Order	20
Confluent Hypergeometric Functions	20
Miscellaneous Functions	20
QUADPACK Background	21
Introduction	21
Survey of Routines	21
Guidelines for the Use of QUADPACK	22
Example Programs	23
Calling Program for QNG	23
Calling Program for QAG	24
Calling Program for QAGS	24
Calling Program for QAGP	25
Calling Program for QAGI	25
Calling Program for QAWO	26
Calling Program for QAWF	26
Calling Program for QAWS	27
Calling Program for QAWC	27
PCHIP: Piecewise Cubic Hermite Interpolation Package	29
PCHIP Routines	29
References	31

Prologue Format for SLATEC Routines	32
Subject Category Overview	39
SLATEC Routines by Subject Category	41
Category A Arithmetic, Error Analysis	41
Category C Elementary and Special Functions (Bessel, Gamma)	41
Category D1A Elementary Vector Operations	45
Category D1B Elementary Matrix Operations	46
Category D2 Solution of Systems of Linear Equations	48
Category D3 Determinants	50
Category D4 Eigenvalues and Eigenvectors (See EISPACK)	50
Category D5 QR Decomposition	51
Category D6 Singular Value Decomposition	51
Category D9 Overdetermined or Underdetermined Systems, Pseudo-inverses	51
Category E Interpolation	52
Category F Solution of Nonlinear Equations	54
Category G Optimization	54
Category H2 Quadrature (Numerical Evaluation of Definite Integrals)	54
Category I1 Ordinary Differential Equations	57
Category I2 Partial Differential Equations	58
Category J1 Fast Fourier Transform	59
Category K Approximation	60
Category L Statistics and Probability	61
Category N Data Handling (I/O, Sorting)	61
Category R1 Machine Constants	61
Category R3 Diagnostics and Error Handling	62
Category Z Other (Documentation)	62
LINPACK Routines	63
Cholesky Operations	63
General Band Matrices	63
General Matrices	63
General Tridiagonal Matrices	63
Hermitian Positive Definite Band Matrices	64
Hermitian Positive Definite Matrices	64
Positive Definite Tridiagonal Matrices	65
Symmetric Matrices	65
Triangular Matrices	65
Complex Hermitian Matrices	66
EISPACK Routines	66
SLATEC Subroutine Dictionary	71
Disclaimer	87
Keyword Index	88
Date and Revisions	90

Preface

- Scope: This document provides usage background on and a function-oriented survey of the subroutines in the SLATEC mathematical library, version 4.1. The other SLATEC manuals contain alphabetically arranged descriptions of individual routines.
- Availability: The SLATEC library is downloadable through LINMath (URL: <http://www.llnl.gov/LCdocs/nmg1>) and can be run on all LC production computers.
- Consultant: For help contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, secure e-mail: lc-hotline@pop.scf.cln).
- Printing: The print file for this document can be found at:
- on the OCF: <http://www.llnl.gov/LCdocs/slatec1/slatec1.pdf>
on the SCF: https://lc.llnl.gov/LCdocs/slatec1/slatec1_scf.pdf

Background on the SLATEC Library

The acronym SLATEC stands for the Sandia, Los Alamos, Air Force Weapons Laboratory Technical Exchange Committee, an organization formed by the computer centers of those three laboratories in New Mexico. Several years later, the National Magnetic Fusion Energy Computer Center, the Livermore Computer Center, and the Sandia National Laboratory Livermore Computer Center became members. The Committee formed several subcommittees to deal with special topics in computing. The SLATEC Mathematical Library Subcommittee is one of those subcommittees. This subcommittee also includes members from Oak Ridge National Laboratory and the National Bureau of Standards, because they were interested in participating in the development of the SLATEC Common Mathematical Library.

The SLATEC library is called common because it will be made available at all the participating sites, to facilitate transfer of applications codes from one site to another. In order to be generally useful, the library must have most of the commonly needed kinds of subprograms. Accordingly, the SLATEC library contains not only routines developed by the participants, but also many routines that are already in the public domain. Examples of the latter are the Basic Linear Algebra Subprograms (BLAS), EISPACK, LINPACK, FFTPACK, the Swarztrauber and Sweet Poisson solvers, and the De Boor B-spline routines. The subcommittee has also arranged to have a specially modified set of the QUADPACK routines released for the SLATEC library.

Using the SLATEC Library

Over 1600 pages of online documentation describe the 902 user-callable subroutines available in version 4.1 of the SLATEC library. Because of this unwieldy bulk, the documentation is published in five separate, but interrelated, volumes:

SLATEC1 (THIS DOCUMENT) provides introductory information on the whole library, explains the subject categories into which the SLATEC routines are grouped, and includes short descriptions of all routines (alphabetical within each subject category). Every category code is also a link (keyword) for retrieving the brief descriptions of the included routines. SLATEC1 provides the only way to compare related routines by the tasks they perform, rather than just by name.

SLATEC2 contains the calling sequence and usage details for each of the 225 subroutines from AAAAAA through D9UPAK, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index.

SLATEC3 contains the calling sequence and usage details for each of the 225 subroutines from DACOSH through DS2Y, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index.

SLATEC4 contains the calling sequence and usage details for each of the 226 subroutines from DSBMV through RD, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index.

SLATEC5 contains the calling sequence and usage details for each of the 226 subroutines from REBAK through ZBIRY, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index.

You can consult any of these documents from any open machine by running your choice of WWW client and selecting the document you want from the descriptive LC collection directory available at . Or you can specifically request the URL

`http://www.llnl.gov/LCdocs/slatecn`

where `slatecn` is any one of `slatec1` through `slatec5`, depending on which volume you want.

Under CTSS, MATHLIB and FORTLIB were dependent libraries of SLATEC; that is, SLATEC had built-in pointers to tell the loader to search MATHLIB and FORTLIB as well. OMNILIB was also a dependent library. Under UNICOS, SLATEC's only dependent library is LIBSCI. Since LIBSCI (or OMNILIB) already contains the BLAS and some of the EISPACK and LINPACK routines, in a form optimized for the CRAY, these (rather than the official SLATEC versions) are used. However, the

documentation for the official SLATEC versions is included in these manuals, since the calling sequence and algorithmic implementation are the same as for the LIBSCI versions.

Before using the SLATEC library, you should at least read the next section, about the SLATEC library error procedure (keyword: [error-procedure](#) (page 8)). You can also retrieve background on the mathematical characteristics of portions of the SLATEC library by using any of these topical links (keywords):

[b-spline-background](#)
[eispack-background](#)
[quadpack-background](#)
[pchip-background](#)

Finally, you can survey the list of available subroutine categories (keyword: [categories](#) (page 39)), or the list of all subroutines sorted alphabetically (keyword: [subroutine-dictionary](#) (page 71)), or a categorized set of subroutine descriptions (keyword: [routines](#) (page 41)) by consulting other sections of SLATEC1 online or in print.

SLATEC Error Procedure

Authors of SLATEC library routines must use at least the first and preferably both of the following techniques to handle errors that their routines detect.

(1) One argument, preferably the last, in the calling sequence must be an error flag if the routine can detect errors. This is an integer variable to which a value is assigned before returning to the caller. A value of zero means the routine completed successfully. A positive value (preferably in the range 1 to 999) should be used to indicate potential, partial, or total failure. Separate values should be used for distinct conditions so that the caller can determine the nature of the failure. Of course, the possible values of this error flag and their meanings must be documented in the description section of the prologue of the routine.

(2) In addition to returning an error flag, the routine can supply more information by writing an error message via a call to XERMSG. XERMSG has an error number as one of its arguments, and the same value that will be returned in the error flag argument must be used in calling XERMSG.

XERMSG is part of the SLATEC Common Math Library error handling package, which consists of a number of routines. It is not necessary for authors to learn about the entire package. Instead we summarize here a few aspects of the package that an author must know (and hence, that a user should look for) in order to use XERMSG correctly.

(1) Although XERMSG supports three levels of severity (warning, recoverable error, and fatal error), be sparing in the use of fatal errors. XERMSG will terminate the program for fatal errors but may return for recoverable errors, and will definitely return after warning messages. An error should be designated fatal only if returning to the caller is likely to be disastrous (e.g. result in an infinite loop).

(2) The error handling package remembers the value of the error number and has an entry point whereby the user can retrieve the most recent error number. Successive calls to XERMSG replace this retained value. In the case of warning messages, it is permissible to issue multiple warnings. In the case of a recoverable error, no additional calls to XERMSG must be made by the Library routine before returning to the caller since the caller must be given a chance to retrieve and clear the error number (and error condition) from the error handling package. In particular, if the user calls Library routine X and X calls a lower level Library Y, it is permissible for Y to call XERMSG, but after it returns to X, X must be careful to note any recoverable errors detected in Y and not make any additional calls to XERMSG in that case.

In practice, it would be simpler if subsidiary routines did not call XERMSG but only returned error flags indicating a serious problem. Then the highest level Library routine could call XERMSG just before returning to its caller. This also allows the highest level routine the most flexibility in assigning error numbers and assures that all possible error conditions are documented in one prologue rather than being distributed through prologues of subsidiary routines.

Each of the arguments to XERMSG is input; none will be modified by XERMSG. A routine may make multiple calls to XERMSG with warning level messages; however, after a call to XERMSG with a recoverable error, the routine should return to the user. Do not try to call XERMSG with a second recoverable error after the first recoverable error because the error package saves the error number. The user can retrieve this error number by calling another entry point in the error handling package and then clear the error number when recovering from the error. Calling XERMSG in succession causes the old error number to be overwritten by the latest error number. This is considered harmless for error numbers associated with

warning messages but must not be done for error numbers of serious errors. After a call to XERMSG with a recoverable error, the user must be given a chance to call NUMXER or XERCLR to retrieve or clear the error number.

B-Splines

Abstract

This section, by Donald E. Amos, describes a B-spline, and the routines necessary to manipulate them at a fairly high level. The basic package described herein is that of Reference 5 (with names altered to prevent duplication and conflicts with routines from Reference 3). The call lists used here are also different. Work vectors were added to ensure portability and proper execution in an overlay environment. These work arrays can be used for other purposes, except as noted in BSPVN. While most of the original routines in Reference 5 were restricted to orders 20 or less, this restriction was removed from all routines, except the quadrature routine BSQAD. (See the section on the Differentiation and Integration of B-splines for details.)

The subroutines referred to below are single precision routines. The corresponding double precision versions are also part of the package, and these have been named by prefixing a D in front of the single precision name. For example, BVALU and DBVALU are the single and double precision versions for evaluating a B-spline or any of its derivatives in the B-representation.

Description of B-Splines

A collection of polynomials of fixed degree $K-1$ defined on a subdivision $(X(I), X(I+1))$, $I=1, \dots, M-1$ of (A, B) with $X(1)=A$, $X(M)=B$ is called a B-spline of order K . If the spline has $K-2$ continuous derivatives on (A, B) , then the B-spline is simply called a spline of order K . Each of the $M-1$ polynomial pieces has K coefficients, making a total of $K(M-1)$ parameters. This B-spline and its derivatives have $M-2$ jumps at the subdivision points $X(I)$, $I=2, \dots, M-1$. Continuity requirements at these subdivision points add constraints and reduce the number of free parameters. If a B-spline is continuous at each of the $M-2$ subdivision points, there are $K(M-1)-(M-2)$ free parameters. In addition, if the B-spline has continuous first derivatives, there are $K(M-1)-2(M-2)$ free parameters, etc., until we get to a spline where we have $K(M-1)-(K-1)(M-2) = M+K-2$ free parameters. Thus, the principle is that increasing the continuity of derivatives decreases the number of free parameters, and conversely.

The points at which the polynomials are tied together by the continuity conditions are called knots. If two knots are allowed to come together at some $X(I)$, then we say that we have a knot of multiplicity 2 there, and the knot values are the $X(I)$ value. If the procedure described in the first paragraph of this section is reversed, we find that adding a knot to increase multiplicity increases the number of free parameters. According to the principle just explained, we have thereby introduced a discontinuity in what was the highest continuous derivative at that knot. Thus, the number of free parameters is $N = NU + K - 2$, where NU is the sum of multiplicities at the $X(I)$ values with $X(1)$ and $X(M)$ of multiplicity 1 ($NU = M$ if all knots are simple, i.e., for a spline, all knots have multiplicity 1.) Each knot can have a multiplicity of at most K . A B-spline is commonly written in the B-representation:

$$Y(X) = \text{sum}(A(I) * B(I, X), I=1, N)$$

to show the explicit dependence of the spline on the free parameters or coefficients $A(I)=BCOEF(I)$ and basis functions $B(I, X)$. These basis functions are themselves special B-splines, which are zero except on

(at most) K adjoining intervals where each $B(I,X)$ is positive and, in most cases, hat or bell-shaped. In order for the nonzero part of $B(I,X)$ to be a spline covering $(X(1),X(2))$, it is necessary to put $K-1$ knots to the left of A , and similarly for $B(N,X)$ to the right of B . Thus, the total number of knots for this representation is $N+2K-2 = N+K$. These knots are carried in an array $T(*)$ dimensioned by at least $N+K$. From the construction, $A=T(K)$ and $B=T(N+1)$ and the spline is defined on $T(K).LE.X.LE.T(N+1)$. The non-zero part of each basis function lies in the Interval $(T(I),T(I+K))$. In many problems where extrapolation beyond A or B is not anticipated, it is common practice to set $T(1)=T(2)=\dots=T(K)=A$ and $T(N+1)=T(N+2)=\dots=T(N+K)=B$.

In summary, since $T(K)$ and $T(N+1)$ as well as interior knots can have multiplicity K , the number of free parameters, $N = \text{sum of multiplicities} - K$. The fact that each $B(I,X)$ function is nonzero over (at most) K intervals, means that for a given X value, there are at most K nonzero terms of the sum. This leads to banded matrices in linear algebra problems, and References 3 and 6 take advantage of this in constructing higher-level routines to achieve speed and avoid ill-conditioning.

Basic Routines

The basic routines that most casual users will need, are those concerned with direct evaluation of splines or B-splines. Since the B-representation, denoted by $(T,BCOEF,N,K)$, is preferred because of numerical stability, the knots $T(*)$, the B-spline coefficients $BCOEF(*)$, the number of coefficients N , and the order K of the polynomial pieces (of degree $K-1$) are usually given. While the knot array runs from $T(1)$ to $T(N+K)$, the B-spline is normally defined on the interval $T(K).LE.X.LE.T(N+1)$. To evaluate the B-spline or any of its derivatives on this interval, one can use:

$$Y = BVALU(T,BCOEF,N,K,X,ID,INBV,WORK)$$

where ID is an integer for the ID -th derivative, $0.LE.ID.LE.K-1$. $ID=0$ gives the zero-th derivative or B-spline value at X . If $X.LT.T(K)$ or $X.GT.T(N+1)$, whether by mistake or the result of round off accumulation in incrementing X , $BVALU$ gives a diagnostic. $INBV$ is an initialization parameter, which is set to 1 on the first call. Distinct splines require distinct $INBV$ parameters. $WORK$ is a scratch vector, with a length of at least $3*K$.

When more conventional communication is needed for publication, or physical interpretation, the B-spline coefficients can be converted to piecewise polynomial (PP) coefficients. Thus, the breakpoints (distinct knots) $XI(*)$, the number of polynomial pieces LXI , and the (right) derivatives $C(*,J)$ at each breakpoint $XI(J)$ are needed to define the Taylor expansion to the right of $XI(J)$ on each interval $XI(J).LE.X.LT.XI(J+1)$, $J=1,LXI$ where $XI(1)=A$ and $XI(LXI+1)=B$. These are obtained from the $(T,BCOEF,N,K)$ representation by:

$$CALL BSPPP(T,BCOEF,N,K,LDC,C,XI,LXI,WORK)$$

where $LDC.GE.K$ is the leading dimension of the matrix C , and $WORK$ is a scratch vector with a length of at least $K*(N+3)$. Then the PP-representation (C,XI,LXI,K) of $Y(X)$, denoted by $Y(J,X)$ on each interval $XI(J).LE.X.LT.XI(J+1)$, is:

$$Y(J,X) = \text{sum}(C(I,J)*((X-XI(J))**(I-1))/\text{factorial}(I-1), I=1,K)$$

for $J=1,\dots,LXI$. One must view this conversion from the B- to the PP-representation with some skepticism, because the conversion may lose significant digits when the B-spline varies in an almost-discontinuous fashion. To evaluate the B-spline or any of its derivatives using the PP-representation, one uses:

$$Y = \text{PPVAL}(LDC,C,XI,LXI,K,ID,X,INPPV)$$

where ID and INPPV have the same meaning and usage as ID and INBV in BVALU.

To determine to what extent the conversion process loses digits, compute the relative error $\text{ABS}((Y1-Y2)/Y2)$ over the X interval with Y1 from PPVAL and Y2 from BVALU. A major reason for considering PPVAL is that evaluation is much faster than with BVALU.

Recall that when multiple knots are encountered, jump-type discontinuities in the B-spline or its derivatives occur at these knots; and we need to know that BVALU and PPVAL return right-limiting values at these knots, except at $X=B$ where left-limiting values are returned. These values are used for the Taylor expansions about the left end points of breakpoint intervals. That is, the derivatives $C(*,J)$ are right derivatives. Note also that a computed X value which, mathematically, would be a knot value may differ from the knot by a round off error. When this happens in evaluating a discontinuous B-spline or some discontinuous derivative, the value at the knot and the value at X can be radically different. In this case, setting X to a T or XI value makes the computation precise. For left-limiting values at knots other than $X=B$, see the prologues to BVALU and other routines.

Interpolation

BINTK is used to generate B-spline parameters (T,BCOEF,N,K), which will interpolate the data by calls to BVALU. A similar interpolation can also be done for cubic splines using BINT4, or the code in Reference 7. If the PP-representation is given, one can evaluate this representation at an appropriate number of abscissas to create data, then use BINTK or BINT4 to generate the B-representation.

Differentiation and Integration

Derivatives of B-splines are obtained from BVALU or PPVAL. Integrals are obtained from BSQAD using the B-representation (T,BCOEF,N,K) and PPQAD using the PP-representation (C,XI,LXI,K). More complicated integrals, involving the product of a function F and some derivative of a B-spline, can be evaluated with BFQAD or PFQAD using the B- or PP-representations, respectively. All quadrature routines, except for PPQAD, are limited in accuracy to 18 digits or working precision, whichever is smaller. PPQAD is limited to working precision only. In addition, the order K for BSQAD is limited to 20 or less. If orders greater than 20 are required, use BFQAD with $F(X) = 1$.

Extrapolation

Extrapolation outside the interval (A,B) can be accomplished easily by the PP-representation, using PPVAL. However, caution should be exercised, especially when several knots are located at A or B, or when the extrapolation is carried significantly beyond A or B. On the other hand, direct evaluation with BVALU outside $A=T(K).LE.X.LE.T(N+1)=B$ produces an error message, and some manipulation of the knots and coefficients is needed to extrapolate with BVALU. This process is described in Reference 6.

Curve Fitting and Smoothing

Unless one has many accurate data points, direct interpolation is not recommended for summarizing data. The results are often not in accordance with intuition, since the fitted curve tends to oscillate through the set of points. Monotone splines (Reference 7) can help curb this undulating tendency but constrained least squares is more likely to give an acceptable fit with fewer parameters. Subroutine FC, described in Reference 6, is recommended for this purpose. The output from this fitting process is the B-representation.

Routines in the B-Spline Package

The subroutines referred to below are single precision. The corresponding double precision versions are also part of the package, and are referenced by prefixing a D in front of the single precision name. For example, BVALU and DBVALU are the single and double precision versions of the routine for evaluating a B-spline or any of its derivatives in the B-representation.

BINT4 - interpolates with splines of order 4
BINTK - interpolates with splines of order k
BSQAD - integrates the B-representation on subintervals
PPQAD - integrates the PP-representation
BFQAD - integrates the product of a function F and any spline derivative in the B-representation
PFQAD - integrates the product of a function F and any spline derivative in the PP-representation
BVALU - evaluates the B-representation or a derivative
PPVAL - evaluates the PP-representation or a derivative
INTRV - gets the largest index of the knot to the left of x
BSPPP - converts from B- to PP-representation
BSPVD - computes nonzero basis functions and derivatives at x
BSPDR - sets up difference array for BSPEV
BSPEV - evaluates the B-representation and derivatives
BSPVN - called by BSPEV, BSPVD, BSPPP and BINTK for function and derivative evaluations

References

1. Computation with Splines and B-Splines, by D. E. Amos, SAND78-1968, March, 1979.
2. Quadrature Subroutines for Splines and B-Splines, by D. E. Amos, SAND79-1825, December, 1979.
3. A Practical Guide to Splines, by C. de Boor, Applied Math. Sci. 27, Springer, N.Y., 1978.
4. On Calculating with B-Splines, by C. de Boor, J. Approx. *SLATECI: Overview and Subject Guide - 13*

- Theory, 6, 50-62 (1972)
5. Package for Calculating with B-Splines, by C. De Boor, SIAM J. Numer. Anal., 14, 441-472 (1977).
 6. Constrained Least Squares Curve Fitting to Discrete Data Using B-Splines - A User's Guide, by R. J. Hanson, SAND78-1291, February, 1979.
 7. Monotone Piecewise Cubic Interpolation, by F. N. Fritsch and R. E. Carlson, LLNL report UCRL-82453, January, 1979.

EISPACK

At this time there is no succinct description of the EISPACK routines. However, each routine is briefly described in the comparative EISPACK catalog included later in this document (keyword: eispack (page 66)).

The reference manual for EISPACK is "Matrix Eigensystem Routines - EISPACK Guide" by B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, published by Springer-Verlag, 1976. This reference describes the original EISPACK, containing many routines, which the user needed to call in the proper order. Subsequently, a number of drivers were written so that the casual user would not have to write all the subroutine calls. The SLATEC library also includes the EISPACK drivers.

Special Functions Background

This section describes the elementary and special function routines in the SLATEC library. Most of them were written by Wayne Fullerton, while he was at LANL. Some were written by Don Amos of SNLA. There are roughly 63 single precision, 63 double precision, and 25 complex, user-callable elementary and special function routines. The table below gives a breakdown of routines according to their function. Unless otherwise indicated, all routines are function subprograms.

Description	Notation	Single Precision	Dble. Prec.	Complex
-------------	----------	------------------	-------------	---------

Intrinsic Functions and Fundamental Functions

Unpack floating point number	Call R9UPAK(X,Y,N)	D9UPAK	--	
Pack floating point number		R9PAK(Y,N)	D9PAK	--
Initialize orthogonal polynomial series	INITS(OS,NOS,ETA)	INITDS	--	
Evaluate Chebyshev series	summation for $i = 1$ to n of $cs(i)*(2*x)**(i-1)$	CSEVL(X,CS,N)	DCSEVL	--

Elementary Functions

Argument = theta in radians	$z = z * e^{i * \theta}$	--	--	CARG(Z)
Cube root		CBRT(X)	DCBRT	CCBRT
Relative error exponential from first order	$((e^{**x}) - 1) / x$	EXPREL(X)	DEXPRL	CEXPRL
Common logarithm	log to the base 10 of z	--	--	CLOG10(Z)
Relative error logarithm	$\ln(1 + x)$	ALNREL(X)	DLNREL	CLNREL
Relative error logarithm from second order	$(\ln(1 + x) - x + x^{**2}/2) / x^{**3}$	R9LN2R(X)	D9LN2R	C9LN2R

Trigonometric and Hyperbolic Functions

Tangent	$\tan z$	--	--	CTAN(Z)
Cotangent	$\cot x$	COT(X)	DCOT	CCOT
Sine x in degrees	$\sin((2*\pi*x)/360)$	SINDG(X)	DSINDG	--
Cosine x in degrees	$\cos((2*\pi*x)/360)$	COSDG(X)	DCOSDG	--
Arc sine	$\arcsin(z)$	--	--	CASIN(Z)
Arc cosine	$\arccos(z)$	--	--	CACOS(Z)
Arc tangent	$\arctan(z)$	--	--	CATAN(Z)
Quadrant correct arc tangent	$\arctan(z1/z2)$	--	--	CATAN2(Z1,Z2)
Hyperbolic sine	$\sinh z$	--	DSINH	CSINH
Hyperbolic cosine	$\cosh z$	--	DCOSH	CCOSH
Hyperbolic tangent	$\tanh z$	--	--	CTANH(Z)
Arc hyperbolic sine	$\operatorname{arcsinh}(x)$	ASINH(X)	DASINH	CASINH
Arc hyperbolic cosine	$\operatorname{arccosh}(x)$	ACOSH(X)	DACOSH	CACOSH
Arc hyperbolic tangent	$\operatorname{arctanh}(x)$	ATANH(X)	DATANH	CATANH
Relative error arc	$(\operatorname{arctan}(x) - x)$	R9ATN1(X)	D9ATN1	--

tangent from first order / x**3

Exponential Integrals and Related Functions

Exponential integral	$Ei(x) =$ the integral from -x to infinity of (e**(-t / t))dt	EI(X)	DEI	--
Exponential integral	$E_{sub 1}(x) =$ the integral from x to infinity of (e**(-t / t)) dt	E1(X)	DE1	--
Logarithmic integral	$li(x) =$ the integral from 0 to x of (1 / ln t) dt	ALI(X)	DLI	--
Sequences of exponential integrals. M values are computed where k=0,1,...M-1 and n>=1				
Exponential integral	$E_{sub n+k}(x)$ =the integral from 1 to infinity of (e**(-x*t)/t**(n+k))dt	Call EXINT(X, N,KODE,M,TOL, EN,IERR)	DEXINT	--

Gamma Functions and Related Functions

Factorial	n!	FAC(N)	DFAC	--
Binomial	$n!/(m!(n-m)!)$	BINOM(N,M)	DBINOM	--
Gamma	gamma(x)	GAMMA(X)	DGAMMA	CGAMMA
Gamma(x) under and overflow limits		Call GAMLIM(XMIN,XMAX)	DGAMLM	--
Reciprocal gamma	1 / gamma(x)	GAMR(X)	DGAMR	CGAMR
Log abs gamma	ln gamma(x)	ALNGAM(X)	DLNGAM	--
Log gamma	ln gamma(z)	--	--	CLNGAM
Log abs gamma with sign	$g = \ln \gamma(x) $ $s = \text{sign } \gamma(x)$	Call ALGAMS(X, G,S)	DLGAMS	--
Incomplete gamma	$\gamma(a,x) =$ the integral from 0 to x of (t**(a-1) * e**(-t))dt	GAMI(A,X)	DGAMI	--
Complementary incomplete gamma	$\gamma(a,x) =$ the integral from x to infinity of (t**(a-1) * e**(-t))dt	GAMIC(A,X)	DGAMIC	--
Tricomi's incomplete gamma	$\gamma^*(a,x)$ = x**(-a) * incomplete gamma(a,x) / gamma(a)	GAMIT(A,X)	DGAMIT	--
Psi (Digamma)	$\psi(x) = \gamma'(x)$ / gamma(x)	PSI(X)	DPSI	CPSI
Pochhammer's generalized symbol	$(a)_{sub x} = \gamma(a+x)$ / gamma(a)	POCH(A,X)	DPOCH	--
Pochhammer's symbol from first order	$((a)_{sub x - 1}) / x$	POCH1(A,X)	DPOCH1	--

Beta	$b(a,b) = \frac{\Gamma(a) \Gamma(b)}{\Gamma(a+b)}$ = the integral from 0 to 1 of $(t^{a-1} (1-t)^{b-1}) dt$	BETA(A,B)	DBETA	CBETA
Log beta	$\ln b(a,b)$	ALBETA(A,B)	DLBETA	CLBETA
Incomplete beta	$i \text{ sub } x (a,b) = \frac{\int_0^x t^{a-1} (1-t)^{b-1} dt}{b(a,b)}$ = 1 / b(a,b) * the integral from 0 to x of $(t^{a-1} (1-t)^{b-1}) dt$	BETAI(X,A,B)	DBETAI	—
Log gamma correction term when Stirling's approximation is valid	$\ln \Gamma(x) - (\ln(2 * \pi)) / 2 - (x - 1/2) * \ln(x) + x$	R9LGMC(X)	D9LGMC	C9LGMC

Error Functions and Fresnel Integrals

Error function	$\text{erf } x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$	ERF(X)	DERF	--
Complementary error function	$\text{erfc } x = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$	ERFC(X)	DERFC	--
Dawson's function	$F(x) = e^{-x^2} \int_0^x e^{t^2} dt$	DAWS(X)	DDAWS	--

Bessel Functions

Bessel functions of special integer order				
First kind, order zero	$J_0(x)$	BESJ0(X)	DBESJ0	--
First kind, order one	$J_1(x)$	BESJ1(X)	DBESJ1	--
Second kind, order zero	$Y_0(x)$	BESY0(X)	DBESY0	--
Second kind, order one	$Y_1(x)$	BESY1(X)	DBESY1	--
Modified (hyperbolic) Bessel functions of special integer order				
First kind, order zero	$I_0(x)$	BESI0(X)	DBESI0	--
First kind, order one	$I_1(x)$	BESI1(X)	DBESI1	--
Third kind, order zero	$K_0(x)$	BESK0(X)	DBESK0	--
Third kind, order one	$K_1(x)$	BESK1(X)	DBESK1	--
Modified (hyperbolic) Bessel functions of special integer order scaled by an exponential				
First kind, order zero	$e^{-x} I_0(x)$	BESIOE(X)	DBSIOE	--
First kind, order one	$e^{-x} I_1(x)$	BESI1E(X)	DBSI1E	--
Third kind, order zero	$e^{-x} K_0(x)$	BESK0E(X)	DBSK0E	--

Third kind, order one	$e^{*x} * K_{sub\ 1}(x)$	BESK1E(X)	DBSK1E	--
Sequences of Bessel functions of general order.				
N values are computed where $k = 1, 2, \dots, N$ and $v \geq 0$.				
Modified first kind	$I_{sub\ v+k-1}(x)$	Call BESI(X,	DBESI	--
	optional scaling	ALPHA,KODE,N,		
	by $e^{*(-x)}$	Y,NZ)		
First kind	$J_{sub\ v+k-1}(x)$	Call BESJ(X,	DBESJ	--
		ALPHA,N,Y,NZ)		
Second kind	$Y_{sub\ v+k-1}(x)$	Call BESY(X,	DBESY	--
		FNU,N,Y)		
Modified third kind	$K_{sub\ v+k-1}(x)$	Call BESK(X,	DBESK	--
	optional scaling	FNU,KODE,N,Y,		
	by $e^{*(x)}$	NZ)		
Sequences of Bessel functions. N values are computed where				
$I = 0, 1, 2, \dots, N-1$ for $N > 0$ or $I = 0, -1, -2, \dots, N+1$				
for $N < 0$.				
Modified third kind	$K_{sub\ v+i}(x)$	Call BESKS(DBESKS	--
		XNU,X,N,BK)		
Sequences of Bessel functions scaled by an exponential.				
N values are computed where $I = 0, 1, 2, \dots, N-1$				
for $N > 0$ or $I = 0, -1, -2, \dots, N+1$ for $N < 0$.				
Modified third kind	$e^{*x} * K_{sub\ v+i}(x)$	Call BESKES(DBSKES	--
		XNU,X,N,BK)		

Bessel Functions of Fractional Order

Airy functions				
Airy	$Ai(x)$	AI(X)	DAI	--
Bairy	$Bi(x)$	BI(X)	DBI	--
Exponentially scaled Airy functions				
Airy	$Ai(x), x \leq 0$	AIE(X)	DAIE	--
	$\exp(2/3 * x^{*(3/2)})$			
	$* Ai(x), x \geq 0$			
Bairy	$Bi(x), x \leq 0$	BIE(X)	DBIE	--
	$\exp(-2/3 * x^{*(3/2)})$			
	$* Bi(x), x \geq 0$			

Confluent Hypergeometric Functions

Confluent hypergeometric	$U(a,b,x)$	CHU(A,B,X)	DCHU	--
--------------------------	------------	------------	------	----

Miscellaneous Functions

Spence dilogarithm	$s(x) = - \int_0^x \frac{\ln 1-y }{y} dy$	SPENC(X)	DSPENC	--
--------------------	--	----------	--------	----

QUADPACK Background

Introduction

QUADPACK is a FORTRAN subroutine package for the numerical computation of definite one-dimensional integrals. It originated with a joint project of R. Piessens and E. de Doncker (Appl. Math. and Progr. Div.; K. U. Leuven, Belgium), C. Ueberhuber (Inst. Fuer Math.; Techn. U. Wien, Austria), and D. Kahaner (Nat'l. Bur. of Standards; Washington, D.C., U.S.A.).

Documentation routine QPDOC describes the package, in the form it was released from the Appl. Math. and Progr. Div. (Leuven), for incorporation into the SLATEC library, in May, 1981. In addition to a survey of the integrators, some guidelines are given, in order to help the QUADPACK user to select an appropriate routine or a combination of several routines for handling his or her problem.

The detailed description of QPDOC includes demonstrations of how to call the integrators, by using small example calling programs.

For precise guidelines involving the use of each routine in particular, we refer to the extensive introductory comments within each routine.

Survey of Routines

The following list gives an overview of the QUADPACK integrators. The letter D precedes the double precision routine names.

- | | |
|------|---|
| QNG | Is a simple nonadaptive automatic integrator, based on a sequence of rules with an increasing degree of algebraic precision (Patterson, 1968). |
| QAG | Is a simple globally-adaptive integrator, using the strategy of Aind (Piessens, 1973). It is possible to choose between six pairs of Gauss-Kronrod quadrature formulae for the rule evaluation component. These pairs are suitable for handling integration difficulties, due to a strongly oscillating integrand, with a high degree of precision. |
| QAGS | Is an integrator based on globally adaptive interval subdivision in connection with extrapolation (de Doncker, 1978) by the Epsilon algorithm (Wynn, 1956). |
| QAGP | Serves the same purposes as QAGS, but also allows for eventual user-supplied information, i.e. the abscissae of internal singularities, discontinuities and other difficulties of the integrand function. The algorithm is a modification of that in QAGS. |
| QAGI | Handles integration over infinite intervals. The infinite range is mapped onto a finite interval, and then the same strategy as in QAGS is applied. |

- QAWO** Is a routine for the integration of $\text{COS}(\text{OMEGA} * X) * F(X)$ or $\text{SIN}(\text{OMEGA} * X) * F(X)$ over a finite interval (A,B). OMEGA is specified by the user. The rule evaluation component is based on the modified Clenshaw-Curtis technique. An adaptive subdivision scheme is used connected with an extrapolation procedure, which is a modification of that in QAGS, and even provides the means for dealing with singularities in F.
- QAWF** Calculates the Fourier cosine or Fourier sine transform of F(X), for user-supplied interval (A, INFINITY), OMEGA, and F. The procedure of QAWO is used on successive finite intervals, and convergence acceleration by means of the Epsilon algorithm (Wynn, 1956) is applied to the series of the integral contributions.
- QAWS** Integrates $W(X) * F(X)$ over (A,B) with A.LT.B finite, and $W(X) = ((X-A)**ALFA) * ((B-X)**BETA) * V(X)$ where $V(X) = 1$ or $\text{LOG}(X-A)$ or $\text{LOG}(B-X)$ or $\text{LOG}(X-A) * \text{LOG}(B-X)$ and $\text{ALFA.GT.}(-1)$, $\text{BETA.GT.}(-1)$. The user specifies A, B, ALFA, BETA, and the type of the function V. A globally adaptive subdivision strategy is applied, with modified Clenshaw-Curtis integration on the subintervals containing A or B.
- QAWC** Computes the Cauchy Principal Value of $F(X)/(X-C)$ over a finite interval (A,B), and for a user-determined C. The strategy is globally adaptive, and modified Clenshaw-Curtis integration is used on the subranges containing the point $X = C$.

Guidelines for the Use of QUADPACK

In this document, we will not investigate the question of when automatic quadrature should be used. Instead, we wish to help those users who have already chosen QUADPACK, by helping them to select an appropriate routine (or combination of routines) for handling their problems.

For quadrature, over both finite and infinite intervals, one of the first questions to be answered by users is related to the amount of computer time they want to spend, versus the time that would be needed, for example, to manually subdivide the interval, or for other analytic manipulations.

(1) The user may not care about computer time, or might not be willing to do any analysis of the problem. This attitude can be perfectly reasonable, especially when only one or a few integrals must be calculated. In this case, it is clear that either the most sophisticated of the routines for finite intervals, QAGS, must be used; or its analogue for infinite intervals, QAGI. These routines are able to cope with rather difficult, and even with improper, integrals.

This way of proceeding may be expensive. But the integrator is supposed to give you an answer in return, with additional information in case of a failure, through its error estimate and flag. Yet, it must be stressed that the programs cannot be totally reliable.

(2) The user may want to examine the integrand function. If bad local difficulties occur at one or more points within the interval, such as: a discontinuity, a singularity, a derivative singularity, or a high peak; our first advice is to split up the interval at these points. The integrand must then be examined separately, over each of the subintervals, so that a suitable integrator can be selected for each one. If problems concerning the relative accuracies to be imposed on finite subintervals result, one can make use of QAGP,

which must be provided with the positions of the local difficulties. However, if strong singularities are present and a high accuracy is requested, applying QAGS to the subintervals may yield a better result.

For quadrature over finite intervals, we can thus dispose of QAGS and:

```
QNG - for well-behaved integrands;
QAG - for functions with an oscillating behaviour of a
      non specific type;
QAWO - for functions, eventually singular, containing
        a factor COS(OMEGA*X) or SIN(OMEGA*X), where OMEGA
        is known;
QAWS - for integrands with Algebraico-Logarithmic end-point
        singularities of known type;
QAWC - for Cauchy Principal Values.
```

On return, the work arrays in the argument lists of the adaptive integrators contain information about the interval subdivision process; and hence about the integrand behaviour, the end points of the subintervals, the local integral contributions and error estimates, and eventually other characteristics. For this reason, and because of its simple globally-adaptive nature, the routine QAG (in particular) is well-suited for integrand examination. Difficult spots can be located by investigating the error estimates on the subintervals.

For infinite intervals, we provide only one general-purpose routine, QAGI. It is based on the QAGS algorithm applied after a transformation of the original interval into (0,1). Yet, it may be that another type of transformation is more appropriate; or one might prefer to break up the original interval and use QAGI only on the infinite part, and so on. These kinds of actions suggest a combined use for different QUADPACK integrators. NOTE: Since QAGI deals with several types of singularity at the boundary point of the integration range, it will not (in general) be necessary to break up the interval when the only difficulty is an integrand singularity. QAGI can also handle slowly convergent improper integrals, if the integrand does not oscillate over the entire infinite interval. If it does, we would advise summing succeeding positive and negative contributions to the integral (e.g. integrate between the zeros) with one or more of the finite-range integrators, and eventually apply convergence acceleration by using the QUADPACK subroutine QELG, which implements the Epsilon algorithm. Such quadrature problems include the Fourier transform as a special case. However, for the latter, we also have the automatic integrator, QAWF, available.

Example Programs

Calling Program for QNG

```
REAL A, ABSERR, B, F, EPSABS, EPSREL, RESULT
INTEGER IER, NEVAL
EXTERNAL F
A = 0.0E0
B = 1.0E0
EPSABS = 0.0E0
EPSREL = 1.0E-3
CALL QNG(F, A, B, EPSABS, EPSREL, RESULT, ABSERR, NEVAL, IER)
C  INCLUDE WRITE STATEMENTS
STOP
```

```

      END
C
      REAL FUNCTION F(X)
      REAL X
      F = EXP(X)/(X*X+0.1E+01)
      RETURN
      END

```

Calling Program for QAG

```

      REAL A, ABSERR, B, EPSABS, EPSREL, F, RESULT, WORK
      INTEGER IER, IWORK, KEY, LAST, LENW, LIMIT, NEVAL
      DIMENSION IWORK(100), WORK(400)
      EXTERNAL F
      A = 0.0E0
      B = 1.0E0
      EPSABS = 0.0E0
      EPSREL = 1.0E-3
      KEY = 6
      LIMIT = 100
      LENW = LIMIT*4
      CALL QAG(F, A, B, EPSABS, EPSREL, KEY, RESULT, ABSERR, NEVAL,
      * IER, LIMIT, LENW, LAST, IWORK, WORK)
C   INCLUDE WRITE STATEMENTS
      STOP
      END
C
      REAL FUNCTION F(X)
      REAL X
      F = 2.0E0/(2.0E0+SIN(31.41592653589793E0*X))
      RETURN
      END

```

Calling Program for QAGS

```

      REAL A, ABSERR, B, EPSABS, EPSREL, F, RESULT, WORK
      INTEGER IER, IWORK, LAST, LENW, LIMIT, NEVAL
      DIMENSION IWORK(100), WORK(400)
      EXTERNAL F
      A = 0.0E0
      B = 1.0E0
      EPSABS = 0.0E0
      EPSREL = 1.0E-3
      LIMIT = 100
      LENW = LIMIT*4
      CALL QAGS(F, A, B, EPSABS, EPSREL, RESULT, ABSERR, IER,
      * LIMIT, LENW, LAST, IWORK, WORK)
C   INCLUDE WRITE STATEMENTS
      STOP
      END
C
      REAL FUNCTION F(X)
      REAL X

```

```

F = 0.0E0
IF(X.GT.0.0E0) F = 1.0E0/SQRT(X)
RETURN
END

```

Calling Program for QAGP

```

REAL A,ABSERR,B,EPSABS,EPSREL,F,POINTS,RESULT,WORK
INTEGER IER,IWORK,LAST,LENIW,LENW,LIMIT,NEVAL,NPTS2
DIMENSION IWORK(204),POINTS(4),WORK(404)
EXTERNAL F
A = 0.0E0
B = 1.0E0
NPTS2 = 4
POINTS(1) = 1.0E0/7.0E0
POINTS(2) = 2.0E0/3.0E0
LIMIT = 100
LENIW = LIMIT*2+NPTS2
LENW = LIMIT*4+NPTS2
CALL QAGP(F,A,B,NPTS2,POINTS,EPSABS,EPSREL,RESULT,ABSERR,
* NEVAL,IER,LENIW,LENW,LAST,IWORK,WORK)
C INCLUDE WRITE STATEMENTS
STOP
END
C
REAL FUNCTION F(X)
REAL X
F = 0.0E+00
IF(X.NE.1.0E0/7.0E0.AND.X.NE.2.0E0/3.0E0) F =
* ABS(X-1.0E0/7.0E0)**(-0.25E0)*
* ABS(X-2.0E0/3.0E0)**(-0.55E0)
RETURN
END

```

Calling Program for QAGI

```

REAL ABSERR,BOUN,EPSABS,EPSREL,F,RESULT,WORK
INTEGER IER,INF,IWORK,LAST,LENW,LIMIT,NEVAL
DIMENSION IWORK(100),WORK(400)
EXTERNAL F
BOUN = 0.0E0
INF = 1
EPSABS = 0.0E0
EPSREL = 1.0E-3
LIMIT = 100
LENW = LIMIT*4
CALL QAGI(F,BOUN,INF,EPSABS,EPSREL,RESULT,ABSERR,NEVAL,
* IER,LIMIT,LENW,LAST,IWORK,WORK)
C INCLUDE WRITE STATEMENTS
STOP
END
C
REAL FUNCTION F(X)

```

```

REAL X
F = 0.0E0
IF(X.GT.0.0E0) F = SQRT(X)*ALOG(X)/
*                ((X+1.0E0)*(X+2.0E0))
RETURN
END

```

Calling Program for QAWO

```

REAL A,ABSERR,B,EPSABS,EPSREL,F,RESULT,OMEGA,WORK
INTEGER IER,INTEGR,IWORK,LAST,LENIW,LENW,LIMIT,MAXP1,NEVAL
DIMENSION IWORK(200),WORK(925)
EXTERNAL F
A = 0.0E0
B = 1.0E0
OMEGA = 10.0E0
INTEGR = 1
EPSABS = 0.0E0
EPSREL = 1.0E-3
LIMIT = 100
LENIW = LIMIT*2
MAXP1 = 21
LENW = LIMIT*4+MAXP1*25
CALL QAWO(F,A,B,OMEGA,INTEGR,EPSABS,EPSREL,RESULT,ABSERR,
* NEVAL,IER,LENIW,MAXP1,LENW,LAST,IWORK,WORK)
C INCLUDE WRITE STATEMENTS
  STOP
  END
C
REAL FUNCTION F(X)
REAL X
F = 0.0E0
IF(X.GT.0.0E0) F = EXP(-X)*ALOG(X)
RETURN
END

```

Calling Program for QAWF

```

REAL A,ABSERR,EPSABS,F,RESULT,OMEGA,WORK
INTEGER IER,INTEGR,IWORK,LAST,LENIW,LENW,LIMIT,LIMLST,
* LST,MAXP1,NEVAL
DIMENSION IWORK(250),WORK(1025)
EXTERNAL F
A = 0.0E0
OMEGA = 8.0E0
INTEGR = 2
EPSABS = 1.0E-3
LIMLST = 50
LIMIT = 100
LENIW = LIMIT*2+LIMLST
MAXP1 = 21
LENW = LENIW*2+MAXP1*25
CALL QAWF(F,A,OMEGA,INTEGR,EPSABS,RESULT,ABSERR,NEVAL,

```

```

      * IER, LIMLST, LST, LENIW, MAXP1, LENW, IWORK, WORK)
C   INCLUDE WRITE STATEMENTS
      STOP
      END
C
      REAL FUNCTION F(X)
      REAL X
      IF(X.GT.0.0E0) F = SIN(50.0E0*X)/(X*SQRT(X))
      RETURN
      END

```

Calling Program for QAWS

```

      REAL A, ABSERR, ALFA, B, BETA, EPSABS, EPSREL, F, RESULT, WORK
      INTEGER IER, INTEGR, IWORK, LAST, LENW, LIMIT, NEVAL
      DIMENSION IWORK(100), WORK(400)
      EXTERNAL F
      A = 0.0E0
      B = 1.0E0
      ALFA = -0.5E0
      BETA = -0.5E0
      INTEGR = 1
      EPSABS = 0.0E0
      EPSREL = 1.0E-3
      LIMIT = 100
      LENW = LIMIT*4
      CALL QAWS(F, A, B, ALFA, BETA, INTEGR, EPSABS, EPSREL, RESULT,
      * ABSERR, NEVAL, IER, LIMIT, LENW, LAST, IWORK, WORK)
C   INCLUDE WRITE STATEMENTS
      STOP
      END
C
      REAL FUNCTION F(X)
      REAL X
      F = SIN(10.0E0*X)
      RETURN
      END

```

Calling Program for QAWC

```

      REAL A, ABSERR, B, C, EPSABS, EPSREL, F, RESULT, WORK
      INTEGER IER, IWORK, LAST, LENW, LIMIT, NEVAL
      DIMENSION IWORK(100), WORK(400)
      EXTERNAL F
      A = -1.0E0
      B = 1.0E0
      C = 0.5E0
      EPSABS = 0.0E0
      EPSREL = 1.0E-3
      LIMIT = 100
      LENW = LIMIT*4
      CALL QAWC(F, A, B, C, EPSABS, EPSREL, RESULT, ABSERR, NEVAL,
      * IER, LIMIT, LENW, LAST, IWORK, WORK)

```

```
C  INCLUDE WRITE STATEMENTS
    STOP
    END
C
    REAL FUNCTION F(X)
    REAL X
    F = 1.0E0/(X*X+1.0E-4)
    RETURN
    END
```

PCHIP: Piecewise Cubic Hermite Interpolation Package

PCHIP Routines

This section describes PCHIP, a new FORTRAN package for the piecewise cubic Hermite interpolation of data. It features software that produces a monotone and "visually pleasing" interpolant to monotone data. As is demonstrated in Reference 1, such an interpolant may be more reasonable to use than a cubic spline, if the data contain both "steep" and "flat" sections. Interpolation of cumulative probability distribution functions is another application. (See References 1 - 3 for examples.)

All piecewise cubic functions in PCHIP are represented in cubic Hermite form. Thus, $F(X)$ is determined by its values $F(I)$ and derivatives $D(I)$ at the breakpoints $X(I)$, $I=1(1)N$.

The routines in PCHIP can be grouped into functional sections, in the following manner:

1. The determination of derivative values.

PCHIM Piecewise Cubic Hermite Interpolation to Monotone data. Use if the data are monotonic, or if you want to guarantee that the interpolant stays within the limits of the data. (See Reference 2.)

PCHIC Piecewise Cubic Hermite Interpolation Coefficients. Use if neither of the above conditions holds, or if you wish control over boundary derivatives. It will generally reproduce monotonicity on subintervals over which the data are monotonic.

PCHSP Piecewise Cubic Hermite Spline. Produces a cubic spline interpolator in cubic Hermite form. Provided primarily for easy comparison of the spline with other piecewise cubic interpolants. (A modified version of de Boor'S CUBSPL, Reference 4.)

2. To evaluate, differentiate, or integrate the resulting piecewise cubic Hermite function. NOTE: If derivative values are available from some other source, these routines can be used without calling any of any of the previous routines.

CHFEV Cubic Hermite Function Evaluator. Evaluates a single cubic Hermite function at an array of points. Used when the interval is known, as in graphing applications. Called by PCHFE.

PCHFE Piecewise Cubic Hermite Function Evaluator. Used when the interval is unknown or the evaluation array spans more than one data interval.

CHFDDV Cubic Hermite Function and Derivative evaluator. Evaluates a single cubic Hermite function and its first derivative at an array of points. Used when the interval is known, as in graphing applications. Called by PCHFD.

PCHFD Piecewise Cubic Hermite Function and Derivative evaluator. Used when the interval is unknown or the evaluation array spans more than one data interval.

PCHID Piecewise Cubic Hermite Integrator, Data limits. Computes the definite integral of a piecewise cubic Hermite function when the integration limits are data points.

PCHIA Piecewise Cubic Hermite Integrator, Arbitrary limits. Computes the definite integral of a piecewise cubic Hermite function over an arbitrary finite interval.

3. Check for monotonicity.

PCHMC Piecewise Cubic Hermite Monotonicity Checker.

4. Internal routines.

CHFIV Cubic Hermite Function Integral eValuator. (Real function called by PCHIA.)

CHFMC Cubic Hermite Function Monotonicity Checker. (Integer function called by PCHMC.)

PCHCE PCHIC End derivative setter. (Called by PCHIC.)

PCHCI PCHIC Initial derivative setter. (Called by PCHIC.)

PCHCS PCHIC Monotonicity Switch derivative setter. (Called by PCHIC.)

PCHDF PCHIP finite Difference Formula. (Real function called by PCHCE and PCHSP.)

PCHST PCHIP Sign Testing routine. (Real function called by various PCHIP routines.)

PCHSW PCHCS SWitch excursion adjuster. (Called by PCHCS.)

The calling sequences for these routines are described in the prologues of the respective routines.

To facilitate two-dimensional applications, the representation of a PCH function throughout the package includes INCFD, the increment between successive elements in the F- and D-arrays. For "normal" usage INCFD=1, and F and D are one-dimensional arrays. The user would call PCHxx (where xx is IM, IC, or SP) with

```
N, X, F, D, 1 .
```

However, suppose that you have data on a rectangular mesh,

```
F2D(I,J) = value at (X(I), Y(J)), I=1(1)NX,  
J=1(1)NY.
```

Assume the following dimensions:

```
REAL X(NXMAX), Y(NYMAX)  
REAL F2D(NXMAX,NYMAX), FX(NXMAX,NYMAX), FY(NXMAX,NYMAX)
```

where 2.LE.NX.LE.NXMAX AND 2.LE.NY.LE.NYMAX . To interpolate in X along the line Y = Y(J), call PCHxx with:

```
NX, X, F2D(1,J), FX(1,J), 1 .
```

To interpolate along the line $X = X(I)$, call PCHxx with:

```
NY, Y, F2D(I,1), FY(I,1), MXMAX .
```

(NOTE: This example assumes the usual column-wise storage of 2-D arrays in FORTRAN.)

References

1. F. N. Fritsch and R. E. Carlson, "Monotone Piecewise Cubic Interpolation," SIAM J. Numer. Anal. 17, 2 (April, 1980), pp. 238-246.
2. F. N. Fritsch and J. Butland, "A Method for Constructing Local Monotone Piecewise Cubic Interpolants," LLNL report UCRL-87559 (April, 1982). [Submitted to Mathematics of Computation.]
3. F. N. Fritsch, "Piecewise Cubic Hermite Interpolation Package," LLNL report UCRL-87285 (July, 1982). [Poster presented at the SIAM 30th Anniversary Meeting, 19-23 July, 1982.]
4. Carl de Boor, A Practical Guide to Splines, Springer-Verlag (New York, 1978). [Especially Chapter IV, pp. 49-62.]

Prologue Format for SLATEC Routines

Each SLATEC subprogram has a section called a prologue that gives standardized information about the routine. The prologue consists of comment lines only. A subsidiary subprogram is one that is usually called by another SLATEC Library subprogram only and is not meant to be called by a user's routine. The prologue for a user-callable subprogram is more extensive than the prologue for a subsidiary subprogram. The prologue for a user-callable subprogram has up to 14 sections, of which 12 are required and one is required if and only if a common block is present. Several of these sections are optional in subsidiary programs and in the quick check routines. The sections are always in the order described in the table below.

Section	User-callable	Subsidiary	Quick Checks
1. BEGIN PROLOGUE	Required	Required	Required
2. SUBSIDIARY	Not present	Required	Optional
3. PURPOSE	Required	Required	Required
4. LIBRARY SLATEC	Required	Required	Required
5. CATEGORY	Required	Optional	Optional
6. TYPE	Required	Required	Required
7. KEYWORDS	Required	Optional	Optional
8. AUTHOR	Required	Required	Required
9. DESCRIPTION	Required	Optional	Optional
10. SEE ALSO	Optional	Optional	Optional
11. REFERENCES	Required	Optional	Optional
12. ROUTINES CALLED	Required	Required	Required
13. COMMON BLOCKS	Required***	Required***	Required***
14. REVISION HISTORY	Required	Required	Required
15. END PROLOGUE	Required	Required	Required

***Note: The COMMON BLOCKS section appears in a subprogram prologue if and only if the subprogram contains a common block.

In the prologue section descriptions that follow, the caret (^) character is used for emphasis to indicate a required blank character.

1. BEGIN PROLOGUE

This section is a single line that immediately follows the subprogram declaration and its continuation lines. It is

```
C***BEGIN^PROLOGUE^^name
```

where "name" (beginning in column 21) is the name of the subprogram.

2. SUBSIDIARY

This section is the single line

```
C***SUBSIDIARY
```

and indicates the routine in which this appears is not intended to be user-callable.

3. PURPOSE

This section gives one to six lines of information on the purpose of the subprogram. The letters may be in upper or lower case. There are no blank lines in the purpose section; i.e., there are no lines consisting solely of a "C" in column 1. The format for the first line and any continuation lines is

```
C***PURPOSE^^information
C^^^^^^^^^^^^^^more information
```

Information begins in column 14 of the first line and no earlier than column 14 of continuation lines.

4. LIBRARY SLATEC

The section is a single line used to show that the routine is a part of the SLATEC library and, optionally, to indicate other libraries, collections, or packages (sublibraries) of which the routine is a part or from which the routine has been derived. The format is

```
C***LIBRARY^^^SLATEC
      or
C***LIBRARY^^^SLATEC^(sublib1,^sublib2,^...sublibn)
```

The leading left parenthesis is immediately followed by the first member of the list. Each member, except for the last, is immediately followed by a comma and a single blank. The last member is immediately followed by the trailing right parenthesis.

5. CATEGORY

This section is a list of classification system categories to which this subprogram might reasonably be assigned. There must be at least one list item. The first category listed is termed the primary category, and others, if given, should be listed in monotonically decreasing order of importance. Categories must be chosen from the classification scheme listed in Appendix A. The required format for the initial line and any continuation lines is

```
C***CATEGORY^^cat1,^cat2,^cat3,^...catn,
C^^^^^^^^^^^^^^continued list
```

All alphabetic characters are in upper case. Items in the list are separated by the two characters, comma and space. If the list will not fit on one line, the line may be ended at a comma (with zero or more trailing spaces), and be continued on the next line. The list and any continuations of the list begin with a nonblank character in column 15.

6. TYPE

This section gives the datatype of the routine and indicates which routines, including itself, are equivalent (except possibly for type) to the routine. The format for this section is

```
C***TYPE^^^^^routine_type^(equivalence list
C^^^^^^^^^^^^^^^^continued equivalence list
C^^^^^^^^^^^^^^^^continued equivalence list)
```

Routine_type, starting in column 15, is the data type of the routine, and is either SINGLE PRECISION, DOUBLE PRECISION, COMPLEX, INTEGER, CHARACTER, LOGICAL, or ALL. ALL is a pseudo-type given to routines that could not reasonably be converted to some other type. Their purpose is typeless. An example would be the SLATEC routine that prints error messages. Equivalence list is a list of the routines (including this one) that are equivalent to this one, but perhaps of a different type. Each item in the list consists of a routine name followed by the "-" character and then followed by the first letter of the type (except use "H" for type CHARACTER) of the equivalent routine. The order of the items is S, D, C, I, H, L and A. The initial item in the list is immediately preceded by a blank and a left parenthesis and the final item is immediately followed by a right parenthesis. Items in the list are separated by the two characters, comma and space. If the list will not fit on one line, the line may be ended at a comma (with zero or more trailing spaces), and be continued on the next line. The list and any continuations of the list begin with a nonblank character in column 15. All alphabetic characters in this section are in upper case. Example

```
C***TYPE          SINGLE PRECISION (ACOSH-S, DACOSH-D, CACOSH-C)
```

7. KEYWORDS

This section gives keywords or keyphrases that can be used by information retrieval systems to identify subprograms that pertain to the topic suggested by the keywords. There must be at least one keyword. Keywords can have embedded blanks but may not have leading or trailing blanks. A keyword cannot be continued on the next line; it must be short enough to fit on one line. No keyword can have an embedded comma. Characters are limited to the FORTRAN 77 character set (in particular, no lower case letters). There is no comma after the last keyword in the list. It is suggested that keywords be in either alphabetical order or decreasing order of importance. The format for the initial line and any continuation lines is

```
C***KEYWORDS^^list
C^^^^^^^^^^^^^^^^continued list
```

Items in the list are separated by the two characters, comma and space. If the list will not fit on one line, the line may be ended at a comma (with zero or more trailing spaces), and be continued on the next line. The list and any continuations of the list begin with a nonblank character in column 15.

8. AUTHOR

This required section gives the author's name. There must be at least one author, and there may be coauthors. At least the last name of the author must be given. The first name (or initials) is optional. The company, organization, or affiliation of the author is also optional. The brackets below indicate optional information. Note that if an organization is to be listed, the remainder of the author's name must also be given. If the remainder of the author's name is given, the last name is immediately followed by a comma. If the organization is given, the first name (or initials) is immediately followed by a comma. The remainder of the name and the organization name may have embedded blanks. The remainder of the name may not have embedded commas. This makes it possible for an information retrieval system to count commas to identify the remainder of the name and the name of an organization. Additional information about the author (e.g., address or telephone number) may be given on subsequent lines. The templates used are

```
C***AUTHOR^^last-name[,^first-name[,^(org)]]
C^^^^^^^^^^^^^^more information
C^^^^^^^^^^^^^^more information
.
.
.
C^^^^^^^^^^^^^^last-name[,^first-name[,^(org)]]
C^^^^^^^^^^^^^^more information
.
.
.
```

Each author's name starts in column 13. Continued information starts in column 15.

9. DESCRIPTION

This section is a description giving the program abstract, method used, argument descriptions, dimension information, consultants, etc. The description of the arguments is in exactly the same order in which the arguments appear in the calling sequence. The description section may use standard, 7-bit ASCII graphic characters, i.e., the 94 printing characters plus the blank. Names of subprograms, common blocks, externals, and formal parameters are all in upper case. Names of variables are also in upper case. The first line of this section is "C***DESCRIPTION" starting in column 1. All subsequent lines in this section start with a "C" in column 1 and no character other than a blank in column 2. Lines with only a "C" in column 1 may be used to improve the appearance of the description. A suggested format for the DESCRIPTION section is given in Appendix E.

10. SEE ALSO

This section is used for listing other SLATEC routines whose prologues contain documentation on the routine in which this section appears. The form is

```
C***SEE ALSO^^name, ^name, ^name
```

where each "name" is the name of a user-callable SLATEC CML subprogram whose prologue provides a description of this routine. The names are given as a list (starting in column 15), with successive names separated by a comma and a single blank.

11. REFERENCES

This section is for references. Any of the 94 ASCII printing characters plus the blank may be used. There may be more than one reference. If there are no references, the section will consist of the single line

```
C***REFERENCES^^(NONE)
```

If there are references, they will be in the following format:

```
C***REFERENCES^^reference 1
C^^^^^^^^^^^^^^^^^^continuation of reference 1
.
.
.
C^^^^^^^^^^^^^^^^^^reference 2
C^^^^^^^^^^^^^^^^^^continuation of reference 2
.
.
.
```

Information starts in column 17 of the first line of a reference and no earlier than column 19 of continuation lines. References should be listed in either alphabetical order by last name or order of citation. They should be in upper and lower case, have initials or first names ahead of last names, and (for multiple authors) have "and" ahead of the last author's name instead of just a comma. The first word of the title of journal articles should be capitalized as should all important words in titles of books, pamphlets, research reports, and proceedings. Titles should be given without quotation marks. The names of journals should be spelled out completely, or nearly so, because software users may not be familiar with them. A complete example of a journal reference is:

```
C      F. N. Fritsch and R. E. Carlson, Monotone piecewise
C      cubic interpolation, SIAM Journal on Numerical Ana-
C      lysis, 17 (1980), pp. 238-246.
```

A complete example of a book reference is:

SLATEC1: Overview and Subject Guide - 36

```

C      Carl de Boor, A Practical Guide to Splines, Applied
C      Mathematics Series 27, Springer-Verlag, New York,
C      1978.

```

12. ROUTINES CALLED

This section gives the names of routines in the SLATEC Common Mathematical Library that are either directly referenced or declared in an EXTERNAL statement and passed as an argument to a subprogram. Note that the FORTRAN intrinsics and other formal parameters that represent externals are not listed. A list is always given for routines called; however, if no routine is called, the list will be the single item "(NONE)" where the parentheses are included. If there are genuine items in the list, the items are in alphabetical order. The collating sequence has "0" through "9" first, then "A" through "Z". The format is

```

C***ROUTINES^CALLED^^name, ^name, ^name, ^name,
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^name, ^name, ^name

```

Items in the list are separated by the two characters, comma and space. If the list will not fit on one line, the line may be ended at a comma (with zero or more trailing spaces), and be continued on the next line. The list and any continuations of the list begin with a nonblank character in column 22.

13. COMMON BLOCKS

This section, that may or may not be required, tells what common blocks are used by this subprogram. If this subprogram uses no common blocks, this section does not appear. If this subprogram does use common blocks, this section must appear. The list of common blocks is in exactly the same format as the list of routines called and uses the same collating sequence. In addition, the name of blank common is "(BLANK)" where the parentheses are included. Blank common should be last in the list if it appears. The format for this section is

```

C***COMMON^BLOCKS^^^^name, ^name, ^name, ^name,
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^name, ^name, ^name^

```

The list starts in column 22.

14. REVISION HISTORY

This section provides a summary of the revisions made to this code. Revision dates and brief reasons for revisions are given. The format is

```

C***REVISION^HISTORY^^(YYMMDD)
C^^^yymmdd^^DATE^WRITTEN
C^^^yymmdd^^revision description

```

```

C^^^^^^^^^^^^^^more revision description
C^^^^^^^^^^^^^^...
C^^^yymmdd^^revision description
C^^^^^^^^^^^^^^more revision description
C^^^^^^^^^^^^^^...
C^^^^^^^^^^^^^^...

```

where, for each revision, "yy" (starting in column 5) is the last two digits of the year, "mm" is the month (01, 02, ..., 12), and "dd" is the day of the month (01, 02, ..., 31). Because this ANSI standard form for the date may not be familiar to some people, the character string "(YYMMDD)" (starting in column 23) is included in the first line of the section to assist in interpreting the sequence of digits. Each line of the revision descriptions starts in column 13. The second line of this section contains the date the routine was written, with the characters "DATE WRITTEN" beginning in column 13. These items must be in chronological order.

15. END PROLOGUE

The last section is the single line

```
C***END^PROLOGUE^^name
```

where "name" is the name of the subprogram.

Subject Category Overview

These are the subject categories (a subset of the GAMS Classification Scheme) used to group the SLATEC subroutines by function performed.

To see an annotated list of the subroutines in any specific category, consult the next section (SLATEC Routines by Subject Category, keyword: [routines](#) (page 41)) or use the category code (such as D6 or H2) as a link to the corresponding subsection. To see detailed documentation for a specific subroutine, consult the alphabetically arranged catalog documents SLATEC2 through SLATEC5. To see an alphabetical list of routine names that indicates the category and function of each routine, consult the SLATEC Subroutine Dictionary below (keyword: [subroutine-dictionary](#) (page 71)).

Category A	Arithmetic, Error Analysis
Category C	Elementary and Special Functions (Gamma, Bessel)
Category D1a	Elementary Vector Operations
Category D1b	Elementary Matrix Operations
Category D2	Solution of Systems of Linear Equations
Category D3	Determinants and Matrices (also use LINPACK)
Category D4	Eigenvalues and Eigenvectors (also use EISPACK)
Category D5	QR Decomposition
Category D6	Singular Value Decomposition
Category D9	Overdetermined or Underdetermined Systems, Pseudo-Inverses
Category E	Interpolation
Category F	Zeros of Functions/Solution of Nonlinear Equations
Category G	Optimization
Category H2	Quadrature (Numerical Evaluation of Definite Integrals)
Category I1	Ordinary Differential Equations
Category I2	Partial Differential Equations
Category J1	Fast Fourier Transforms
Category K	Approximation
Category L	Statistics, Probability, Random Numbers
Category N	Data Handling (I/O, Sorting)
Category R1	Machine-dependent Constants

Category R3 Error Handling

Category Z Other

SLATEC Routines by Subject Category

Category A Arithmetic, Error Analysis

single	double	complx	
-----	-----	-----	
R9PAK	D9PAK	-	Pack a base 2 exponent into a floating point number.
R9UPAK	D9UPAK	-	Unpack a floating point number X so that $X=Y*2**N$.
XADD	DXADD	-	Provide single- (or double-) precision floating-point arithmetic with an extended exponent range.
XADJ	DXADJ	-	Provide single- (or double-) precision floating-point arithmetic with an extended exponent range.
XC210	DXC210	-	Provide single- (or double-) precision floating-point arithmetic with an extended exponent range.
XCON	DXCON	-	Provide single- (or double-) precision floating-point arithmetic with an extended exponent range.
XRED	DXRED	-	Provide single- (or double-) precision floating-point arithmetic with an extended exponent range.
XSET	DXSET	-	Provide single- (or double-) precision floating-point arithmetic with an extended exponent range.

Category C Elementary and Special Functions (Bessel, Gamma)

single	double	complx	
-----	-----	-----	
-	-	CACOS	Compute the complex arc Cosine.
-	-	CARG	Compute the argument of a complex number.
-	-	CASIN	Compute the complex arc Sine.
-	-	CATAN	Compute the complex arc Tangent.
-	-	CATAN2	Compute the complex arc Tangent in the proper quadrant.
-	-	CCOSH	Compute the complex hyperbolic Cosine.
-	-	CLOG10	Compute the principal value of the complex base 10 log.
-	-	CSINH	Compute the complex hyperbolic Sine.
-	-	CTAN	Compute the complex Tangent.
-	-	CTANH	Compute the complex hyperbolic Tangent.
ACOSH	DACOSH	CACOSH	Compute the arc hyperbolic Cosine.

AI	DAI	-	Compute the Airy function.
AIE	DAIE	-	Compute the exponentially scaled Airy function.
-	ZAIRY	CAIRY	Compute the complex Airy function $A_i(z)$ or its derivative dA_i/dz .
ALBETA	DLBETA	CLBETA	Compute the natural log of the complete Beta function.
ALGAMS	DLGAMS	-	Compute the log of the absolute value of the Gamma function including the sign.
ALI	DLI	-	Compute the logarithmic integral.
ALNGAM	DLNGAM	CLNGAM	Compute the log of the absolute value of the Gamma function.
ALNREL	DLNREL	CLNREL	Compute $\ln(1+X)$ accurate in the sense of relative error.
ASINH	DASINH	CASINH	Compute the arc hyperbolic Sine.
ATANH	DATANH	CATANH	Compute the arc hyperbolic Tangent.
-	ZBESH	CBESH	Compute an N member sequence of complex Hankel (Bessel) functions.
BESI	DBESI	-	Compute an N member sequence of Bessel functions $I_{\text{sub}}(\text{ALPHA}+K-1)$ at X , $K=1, \dots, N$ or scaled Bessel functions $\text{EXP}(-X) * I_{\text{sub}}(\text{ALPHA}+K-1)$ at X , $K=1, \dots, N$ for non-negative ALPHA and X.
-	ZBESI	CBESI	Compute an N member sequence of complex Bessel functions $I(a, z)$.
BESI0	DBESI0	-	Compute the hyperbolic Bessel function of the first kind of order zero.
BESI0E	DBESI0E	-	Compute the exponentially scaled hyperbolic Bessel function of the first kind of order zero.
BESI1	DBESI1	-	Compute the hyperbolic Bessel function of first kind of order one.
BESI1E	DBESI1E	-	Compute the exponentially scaled hyperbolic Bessel function of the first kind of order one.
BESJ	DBESJ	-	Compute an N member sequence of J Bessel functions $J_{\text{sub}}(\text{ALPHA}+K-1)$ at X , $K=1, \dots, N$ for non-negative ALPHA and X.
-	ZBESJ	CBESJ	Compute an N member sequence of complex Bessel functions $J(a, z)$.
BESJ0	DBESJ0	-	Compute the Bessel function of the first kind of order zero.
BESJ1	DBESJ1	-	Compute the Bessel function of the first kind of order one.
BESK	DBESK	-	Implements forward recursion on the three term recursion

relation for a sequence of non-negative order Bessel functions $K_{\text{sub}(FNU+I-1)}$, or scaled Bessel functions $EXP(X)*K_{\text{sub}(FNU+I-1)}$ at $X, I=1, \dots, N$ for $X .gt. 0.0E0$ and non-negative orders FNU .

- ZBESK CBESK Compute an N member sequence of complex Bessel functions $K(a, z)$.
- BESK0 DBESK0 - Compute the hyperbolic Bessel function of the third kind of order zero.
- BESK0E DBSK0E - Compute the exponentially scaled hyperbolic Bessel function of the third kind of order zero.
- BESK1 DBESK1 - Compute the hyperbolic Bessel function of the third kind of order one.
- BESK1E DBSK1E - Compute the exponentially scaled hyperbolic Bessel function of the third kind of order one.
- BESKES DBSKES - Compute a sequence of exponentially scaled modified Bessel functions of the third kind of fractional order.
- BESKS DBESKS - Compute a sequence of modified Bessel functions of the third kind of fractional order.
- BESY DBESY - Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y_{\text{sub}(FNU+I-1)}$ at $X, I=1, N$ for real $X .gt. 0$ and for non-negative orders FNU .
- ZBESY CBESY Compute an N member sequence of complex Bessel functions $Y(a, z)$.
- BESY0 DBESY0 - Compute the Bessel function of the second kind of order zero.
- BESY1 DBESY1 - Compute the Bessel function of the second kind of order one.
- BETA DBETA CBETA Compute the complete Beta function.
- BETAI DBETAI - Compute the incomplete Beta function.
- BI DBI - Compute the Bairy function (Airy, second kind).
- BIE DBIE - Compute the exponentially scaled Bairy function.
- ZBIRY CBIRY Compute the complex Bairy function $B_i(z)$ or its derivative dB_i/dz .
- BINOM DBINOM - Compute the binomial coefficients.
- BSKIN DBSKIN - Compute repeated integrals of the K-zero Bessel function.
- CBRT DCBRT CCBRT Compute the cube root of the argument.
- CHU DCHU - Compute the logarithmic confluent hypergeometric function.

COSDG	DCOSDG	-	Compute the Cosine of an argument in degrees.
COT	DCOT	CCOT	Compute the complex Cotangent.
CSEVL	DCSEVL	-	Evaluate the N-term Chebyshev series.
DAWS	DDAWS	-	Compute Dawson's function.
E1	DE1	-	Compute the exponential integral E1(X).
EI	DEI	-	Compute the exponential integral EI(X).
ERF	DERF	-	Compute the error function.
ERFC	DERFC	-	Compute the complementary error function.
EXINT	DEXINT	-	Compute M member sequences of exponential integrals $E(N+K, X)$, $K=0, 1, \dots, M-1$ for $N \geq 1$ and $X \geq 0$.
EXPREL	DEXPRL	CEXPRL	Evaluate $EXPREL(X) = (EXP(X)-1)/X$.
FAC	DFAC	-	Compute the factorial of N.
FUNDOC	-	-	Documentation for FNLIB, routines for evaluating elementary and special functions.
GAMI	DGAMI	-	Compute the incomplete Gamma function.
GAMIC	DGAMIC	-	Compute the complementary incomplete Gamma function.
GAMIT	DGAMIT	-	Compute Tricomi's form of the incomplete Gamma function.
GAMLIM	DGMLM	-	Compute the minimum and maximum bounds for GAMMA.
GAMMA	DGAMMA	CGAMMA	Compute the complete Gamma function.
GAMR	DGAMR	CGAMR	Compute the reciprocal Gamma function.
INITS	INITDS	-	Initialize to determine the number of terms to carry in an orthogonal series to meet a specified error.
POCH	DPOCH	-	Compute a generalization of Pochhammer's symbol.
POCH1	DPOCH1	-	Compute a generalization of Pochhammer's symbol starting from first order.
PSI	DPSI	CPSI	Compute the Psi (or Digamma) function.
PSIFN	DPSIFN	-	Compute derivatives of the Psi function.
-	-	COLGMC	Evaluates $(z+0.5)*CLOG((Z+1.)/Z) - 1.0$ with relative accuracy.
RC	DRC	-	Compute the Elliptic integral defined as the integral from zero to infinity of $.5*dt/((t+Y)*sqrt(t+X))$.
RC3JJ	DRC3JJ	-	Evaluate the 3J symbol $f(L1)$ for all allowed values of L1.

RC3JM	DRC3JM	-	Evaluate the 3J symbol $g(M2)$ for all allowed values of $M2$.
RC6J	DRC6J	-	Evaluate the 6J symbol $h(L1)$ for all allowed values of $L1$.
RD	DRD	-	Compute the incomplete or complete Elliptic integral of the second kind.
RF	DRF	-	Compute the incomplete or complete Elliptic integral of the first kind.
RJ	DRJ	-	Compute the incomplete or complete Elliptic integral of the third kind.
SINDG	DSINDG	-	Compute the Sine of an argument in degrees.
SPENC	DSPENC	-	Compute a form of Spence's integral due to K. Mitchell.
XLEGF	DXLEGF	-	Compute the values of Legendre functions using extended-range arithmetic.
XNRMP	DXNRMP	-	Compute normalized Legendre polynomials of varying order but fixed argument and degree.

Category D1A Elementary Vector Operations

single double complx

ISAMAX	IDAMAX	ICAMAX	Find the largest component of a vector.
SASUM	DASUM	SCASUM	Sum of the magnitudes of vector components.
SAXPY	DAXPY	CAXPY	Computation (for scalar a) of $y = a*x + y$.
SCOPY	DCOPY	CCOPY	(Also ICOPY) vector copy $y = x$.
SCOPYM	DCOPYM	-	Vector copy (with sign change) $y = -x$.
SDOT	DDOT	CDOTU	Vector inner (dot) product.
-	-	CDOTC	Dot product of complex vectors using complex conjugate of the first vector.
SDSDOT	-	CDCDOT	Vector dot product with double precision accumulation.
-	DSDOT	DCDOT	Double precision dot product of single precision vectors
SNRM2	DNRM2	SCNRM2	Euclidean length (L2 norm) of a vector.
SROT	DROT	CSROT	Apply a plane Given's rotation.
SROTG	DROTG	CROTG	Construct a plane Given's rotation.
SROTM	DROTM	-	Apply a modified Given's transformation.

SROTMG	DROTMG	-	Construct a modified Given's transformation.
SSCAL	DSCAL	CSCAL	Vector scale $x = a*x$.
-	-	CSSCAL	Scale a complex vector.
SSWAP	DSWAP	CSWAP	ISWAP Interchange two vectors.
-	DQDOTA	-	Inner product with extended precision accumulation and result.
-	DQDOTI	-	Inner product with extended precision accumulation and result.

Category D1B Elementary Matrix Operations

single	double	complx
-----	-----	-----

SGEMM	DGEMM	CGEMM	Multiply a real general matrix by a real general matrix.
SGEMV	DGEMV	CGEMV	Multiply a real vector by a real general matrix.
SGER	DGER	-	Perform a rank 1 update of a real general matrix.
-	-	CGERC	Perform conjugated rank 1 update of a complex general matrix.
-	-	CGERU	Perform unconjugated rank 1 update of a complex general matrix.
-	-	CHBMV	Multiply a complex vector by a complex Hermitian band matrix.
-	-	CHEMM	Multiply a complex general matrix by a complex Hermitian matrix.
-	-	CHEMV	Multiply a complex vector by a complex Hermitian matrix.
-	-	CHER	Perform Hermitian rank 1 update of a complex Hermitian matrix.
-	-	CHER2	Perform Hermitian rank 2 update of a complex Hermitian matrix.
-	-	CHER2K	Perform Hermitian rank 2K update of a complex Hermitian matrix.
-	-	CHERK	Perform Hermitian rank K update of a complex Hermitian matrix.
-	-	CHPMV	Multiply a Hermitian matrix by scalars.

-	-	CHPR	Perform the Hermitian rank 1 operation $A := \alpha * X * \text{conj}(X') + A.$
-	-	CHPR2	Perform the Hermitian rank 2 operation like CHPR.
SS2Y	DS2Y	-	Convert from SLAP Triad to SLAP Column format.
SSBMV	DSBMV	-	Multiply a real vector by a real symmetric band matrix.
SSDI	DSDI	-	Calculate the product $X = \text{DIAG} * B$, where DIAG is a diagonal matrix.
SSMTV	DSMTV	-	Calculate the sparse matrix transpose vector product $Y = A' * X.$
SSMV	DSMV	-	Calculate the sparse matrix vector product $Y = A * X.$
SSPMV	DSPMV	-	Perform the matrix-vector operation $y := \alpha * A * X + \beta * y.$
SSPR	DSPR	-	Perform the symmetric rank 1 operation $A := \alpha * X * X' + A.$
SSPR2	DSPR2	-	Perform the symmetric rank 2 operation $A := \alpha * X * Y' + \alpha * Y * X' + A.$
SSYMM	DSYMM	CSYMM	Multiply a real general matrix by a real symmetric matrix.
SSYMV	DSYMV	-	Multiply a real vector by a real symmetric matrix.
SSYR	DSYR	-	Perform a symmetric rank 1 update of a real symmetric matrix.
SSYR2	DSYR2	-	Perform a symmetric rank 2 update of a real symmetric matrix.
SSYR2K	DSYR2K	CSYR2K	Perform a symmetric rank 2K update of a real symmetric matrix.
SSYRK	DSYRK	CSYRK	Perform a symmetric rank K update of a real symmetric matrix.
STBMV	DTBMV	CTBMV	Multiply a real vector by a real triangular band matrix.
STBSV	DTBSV	CTBSV	Solve a real triangular banded system of linear equations.
STPMV	DTPMV	CTPMV	Perform the matrix-vector operation $X := A * X$ or $X := A' * X.$
STPSV	DTPSV	CTPSV	Solve the system of equations $A * X = b$ or $A' * X = b.$
STRMM	DTRMM	CTRMM	Multiply a real general matrix by a real triangular matrix.

STRMV	DTRMV	CTRMV	Multiply a real vector by a real triangular matrix.
STRSM	DTRSM	CTRSM	Solve a real triangular system of equations with multiple right-hand sides.
STRSV	DTRSV	CTRSV	Solve a real triangular system of linear equations. multiple right-hand sides.

Category D2 Solution of Systems of Linear Equations

Also see the LINPACK routines, , keyword: [linpack](#) (page 63).

single double complx

SBCG	DBC	-	Solve a nonsymmetric linear system using the preconditioned biconjugate gradient method.
SCG	DCG	-	Solve a symmetric positive definite linear system using the preconditioned conjugate gradient method.
SCGN	DCGN	-	Solve a general linear system using the preconditioned conjugate gradient method on normalized equations.
SCGS	DCGS	-	Solve a nonsymmetric linear system using the preconditioned conjugate gradient squared method.
SGEFS	DGEFS	CGEFS	Solve a general real (double,complex) NxN system of linear equations.
SGEIR	-	CGEIR	Solve a general real (complex) NxN system of linear equations. Iterative refinement is used to obtain an error estimate.
SGMRES	DGMRES	-	Solve a nonsymmetric linear system using the generalized minimum residual method.
SIR	DIR	-	Solve a general linear system using iterative refinement with matrix splitting.
SLLT12	DLLT12	-	Use SLAP backsolve for LDL' incomplete factorization.
SLPDOC	DLPDOC	-	Solve large sparse positive definite linear systems using preconditioned iterative methods.
SNBCO	DNBCO	CNBCO	Factor a BAND matrix by Gaussian elimination and estimates the condition of the matrix.
SNBFA	DNBFA	CNBFA	Factor a BAND matrix by Gaussian elimination.
SNBFS	DNBFS	CNBFS	Solve a general nonsymmetric banded real (double,complex) NxN system of linear equations.
SNBIR	-	CNBIR	Solve a general nonsymmetric banded real (complex) NxN system of linear equations. Iterative refinement is used to obtain an error estimate.

SNBSL	DNBSL	CNBSL	Solve a BANDED system using factors created by SNBCO or SNBFA, or by DNBCO or DNBFA, or by CNBCO or CNBFA.
SOMN	DOMN	-	Solve a general linear system using the preconditioned orthomin method.
SPOFS	DPOFS	CPOFS	Solve a positive definite symmetric real (double,complex) NxN system of linear equations.
SPOIR	-	CPOIR	Solve a positive definite real symmetric (complex Hermitian) NxN system of linear equations. Iterative refinement is used to obtain an error estimate.
SS2LT	DS2LT	-	Store the lower triangle of a matrix stored in the SLAP Column format.
SSD2S	DSD2S	-	Compute the inverse of the diagonal of the matrix A*A' where A is in SLAP Column format.
SSDBC	DSDBC	-	Solve a linear system using the biconjugate gradient method with diagonal scaling.
SSDCG	DSDCG	-	Solve a symmetric positive definite linear system using the preconditioned conjugate gradient method.
SSDCGN	DSDCGN	-	Solve a general linear system using the conjugate gradient method with diagonal scaling, applied to the normal equations.
SSDCGS	DSDCGS	-	Solve a linear system using the biconjugate gradient squared method with diagonal scaling.
SSDGMR	DSDGMR	-	Solve a linear system using the generalized minimum residual method with diagonal scaling.
SSDOMN	DSDOMN	-	Solve a general linear system using the Orthomin method with diagonal scaling.
SSDS	DSDS	-	Compute the inverse of the diagonal of a matrix stored in SLAP Column format.
SSDSCL	DSDSCL	-	Scale and unscale Ax = b by symmetric diagonal scaling.
SSGS	DSGS	-	Solve a general linear system using Gauss-Seidel iteration.
SSICGC	DSICGC	-	Solve a symmetric positive definite linear system using the incomplete Cholesky preconditioned conjugate gradient method.
SSICS	DSICS	-	Generate the incomplete Cholesky decomposition of a symmetric positive definite matrix stored in SLAP Column format.
SSILUR	DSILUR	-	Solve a general linear system using the LU decomposition with iterative refinement.
SSILUS	DSILUS	-	Generate the incomplete LDU decomposition

of a matrix.

- | | | | |
|--------|--------|---|---|
| SSJAC | DSJAC | - | Solve a general linear system using Jacobi iteration. |
| SSLI | DSLII | - | Interface for SLAP MSOLVE for lower triangular matrix. |
| SSLI2 | DSLII2 | - | SLAP backsolve $Lx = b$, where L is a lower triangular matrix. |
| SSLITI | DSLITI | - | Interface for SLAP MSOLVE for LDL' (IC) factorization. |
| SSLUBC | DSLUBC | - | Solve a linear system using the biconjugate gradient method with incomplete LU decomposition preconditioning. |
| SSLUCN | DSLUCN | - | Solve a general linear system using the incomplete LU decomposition with the conjugate gradient method applied to normal equations. |
| SSLUCS | DSLUCS | - | Solve a linear system using the biconjugate gradient squared method with incomplete LU decomposition preconditioning. |
| SSLUGM | DSLUGM | - | Solve a linear system using the generalized minimum residual method with incomplete LU factorization for preconditioning. |
| SSLUI | DSLUI | - | Interface for SLAP MSOLVE for LDU factorization. |
| SSLUI2 | DSLUI2 | - | SLAP backsolve $L*D*U x = b$, for LDU factorization. |
| SSLUI4 | DSLUI4 | - | SLAP backsolve $(L*D*U)' x = b$, for LDU factorization. |
| SSLUOM | DSLUOM | - | Solve a general linear system using the Orthomin method with incomplete LU decomposition. |
| SSLUTI | DSLUTI | - | Interface for SLAP MTSOLV for LDU factorization. |
| SSMMI2 | DSMMI2 | - | SLAP backsolve for LDU factorization of normal equations. |
| SSMMTI | DSMMTI | - | Interface for SLAP MMTSLV for LDU factorization of normal equations. |

Category D3 Determinants

single double complex

- | | | | |
|-------|-------|-------|---|
| SNBDI | DNBDI | CNBDI | Compute the determinant of a BAND matrix using factors created by SNBCO or SNBFA, or by DNBCO or DNBFA, or by CNBCO or CNBFA. |
|-------|-------|-------|---|

Category D4 Eigenvalues and Eigenvectors (See EISPACK)

Also see EISPACK routines, , keyword: [eispack](#) (page 66).

a QR factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency.

- | | | | |
|-------|--------|---|---|
| SGLSS | DGLSS | - | Solve linear least squares problems by performing a QR factorization using Householder transformations. Emphasis is put on detecting possible rank deficiency. |
| ULSIA | DULSIA | - | Solves the UNDERDETERMINED LINEAR system of equations by performing an LQ factorization of the matrix A using Householder transformations. Emphasis is put on detecting possible rank deficiency. |

Category E Interpolation

single double complx

- | | | | |
|-------|--------|---|--|
| BFQAD | DBFQAD | - | Compute the integral on (X1,X2) of a product of a function and the ID-th derivative of a K-th order B-spline (B-representation). |
| BINT4 | DBINT4 | - | Compute the B representation of a cubic spline which interpolates data (X(I),Y(I)), I=1,NDATA. |
| BINTK | DBINTK | - | Produce the B-spline coefficients, BCOEF, of the B-spline of order K with knots T(I), I=1,...,N+K, which takes on the value Y(I) at X(I), I=1,...,N. |
| BSPDR | DBSPDR | - | Use the B-representation to construct a divided difference table preparatory to a (right) derivative calculation in BSPEV. |
| BSPEV | DBSPEV | - | Compute the value of the spline and its derivatives from the B-representation. |
| BSPPP | DBSPPP | - | Convert the B-representation to the piecewise polynomial (PP) form for use with PPVAL. |
| BSPVD | DBSPVD | - | Compute the value and all derivatives of order less than NDERIV of all basis functions which do not vanish at X. |
| BSPVN | DBSPVN | - | Compute the value of all (possibly) nonzero basis functions at X. |
| BSQAD | DBSQAD | - | Compute the integral on (X1,X2) of a K-th order B-spline using the B-representation. |
| BVALU | DBVALU | - | Evaluate the B-representation of a B-spline for the function value or any of its derivatives. |
| CHFDV | DCHFDV | - | Evaluate a cubic polynomial given in Hermite form and its first derivative at an array of points. While designed for use by PCHFD, may be used directly as an evaluator in applications, such as graphing, where the interval is known in advance. If only function values are required, use CHFV instead. |

CHFEV	DCHFEV	-	Evaluate a cubic polynomial given in Hermite form at an array of points. While designed for use by PCHFE, may be used directly as an evaluator in applications, such as graphing, where the interval is known in advance.
INTRV	DINTRV	-	Compute the largest integer ILEFT in the interval [1,LXT] such that $XT(ILEFT) \leq X$ where $XT(*)$ is a subdivision of the X interval.
PCHBS	DPCHBS	-	Convert peicewise cubic Hermite to B-spline.
PCHCM	DPCHCM	-	Check a cubic Hermite function for monotonicity.
PCHFD	DPCHFD	-	Evaluate a piecewise cubic Hermite function and its first derivative at an array of points. May be used by itself for Hermite interpolation, or as an evaluator for PCHIM or PCHIC. If only function values are required, use PCHFE instead.
PCHFE	DPCHFE	-	Evaluate a piecewise cubic Hermite function at an array of points. May be used by itself for Hermite interpolation, or as an evaluator for PCHIM or PCHIC.
PCHIA	DPCHIA	-	Evaluate the definite integral of a piecewise cubic Hermite function over an arbitrary interval.
PCHIC	DPCHIC	-	Set derivatives needed to determine a piecewise monotone piecewise cubic Hermite interpolant to given data. User control is available over boundary conditions and/or the treatment of points where monotonicity switches direction.
PCHID	DPCHID	-	Evaluate the definite integral of a piecewise cubic Hermite function over an interval whose endpoints are data points.
PCHIM	DPCHIM	-	Set derivatives needed to determine a monotone piecewise cubic Hermite interpolant to given data. Boundary values are provided which are compatible with monotonicity. The interpolant will have an extremum at each point where monotonicity switches direction. Use PCHIC if control is desired over boundary or switch conditions.
PCHDOC	Documentation for the PCHIP routines. PCHIP is a Fortran package for piecewise cubic Hermite interpolation of data. It features software to produce a monotone and "visually pleasing" interpolant to monotone data.		
PCHSP	DPCHSP	-	Produce a cubic spline interpolator in cubic Hermite form. Provided primarily for easy comparison of the spline with other piecewise cubic interpolants.
PFQAD	DPFQAD	-	Compute the integral of a product of a function and the ID-th derivative of a B-spline, (PP-representation).
POLCOF	DPOLCF	-	Compute the coefficients of polynomial fit (including Hermite) produced by previous call to POLINT.
POLINT	DPLINT	-	Produce the polynomial which interpolates a set of

discrete data points.

- POLYVL DPOLVL - Calculate the value of the polynomial and its first NDER derivatives where the polynomial was produced by a previous call to POLINT.
- PPQAD DPPQAD - Compute the integral on (X1,X2) of a K-th order B-spline using the piecewise polynomial representation.
- PPVAL DPPVAL - Compute the value of the IDERIV-th derivative of the B-spline from the PP-representation.

Category F Solution of Nonlinear Equations

single double complx

- CHKDER DCKDER - Check the gradients of M nonlinear functions in N variables, evaluated at a point X, for consistency with the functions themselves.
- FZERO DFZERO - Find a zero of a function F(X) in a given interval [B,C]. It is designed primarily for problems where F(B) and F(C) have opposite signs.
- RPQR79 - CPQR79 Find the zeros of a polynomial with real (complex) coefficients using the companion matrix method.
- RPZERO - CPZERO Find the zeros of a polynomial with real (complex) coefficients using a modified Newton method.
- SNSQ DNSQ - Find a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. This code is the combination of the MINPACK codes HYBRD and HYBRDJ.
- SNSQE DNSQE - The easy-to-use version of SNSQ. This code is the combination of the MINPACK codes HYBRD1 and HYBRJ1.
- SOS DSOS - Solve a square system of nonlinear equations using a method similar to Brown's method.

Category G Optimization

single double complx

- SPLP DSPLP - Solve linear programming problems involving at most a few thousand constraints and variables. Takes advantage of sparsity in the constraint matrix.

Category H2 Quadrature (Numerical Evaluation of Definite Integrals)

single double complx

AVINT	DAVINT	-	Integrate a function tabulated at arbitrarily spaced abscissas using overlapping parabolas.
GAUS8	DGAUS8	-	Integrate real functions of one variable over finite intervals using an adaptive 8-point Legendre-Gauss algorithm.
QAG	DQAG	-	Estimate a definite integral over a finite interval.
QAGI	DQAGI	-	Estimate a definite integral over a semi-infinite or infinite interval.
QAGP	DQAGP	-	Estimate a definite integral over a finite interval with user supplied break points.
QAGS	DQAGS	-	Estimate a definite integral over a finite interval with extrapolation.
QAWC	DQAWC	-	Estimate a definite integral over a finite interval with a Cauchy weight function.
QAWF	DQAWF	-	Estimate a definite integral over a semi-infinite interval with a Fourier weight function.
QAWO	DQAWO	-	Estimate a definite integral over a finite interval with an oscillatory weight function.
QAWS	DQAWS	-	Estimate a definite integral over a finite interval with an algebraic or logarithmic weight function.
QNC79	DQNC79	-	Integrate a user defined function by a 7-point adaptive Newton-Cotes quadrature rule.
QNG	DQNG	-	Estimate a definite integral over a finite interval using a non-adaptive scheme.
QPDOC	Documentation for the QUADPACK numerical quadrature package.		
QAGE	DQAGE	-	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESLT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
QAGSE	DQAGSE	-	The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
QK15	DQK15	-	To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{integral of } \text{ABS}(F) \text{ over } (A,B)$
QK21	DQK21	-	To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
QK31	DQK31	-	To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
QK41	DQK41	-	To compute $I = \text{Integral of } F \text{ over } (A,B)$, with error

- estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
- QK51 DQK51 - To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$
- QK61 DQK61 - To compute $I = \text{Integral of } F \text{ over } (A,B)$ with error estimate $J = \text{Integral of } \text{DABS}(F) \text{ over } (A,B)$
- QAGPE DQAGPE - Approximate a given definite integral $I = \text{Integral of } F \text{ over } (A,B)$, hopefully satisfying the accuracy claim: $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$. Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities or discontinuities) are provided by the user.
- QAWCE DQAWCE - The routine calculates an approximation result to a CAUCHY PRINCIPAL VALUE $I = \text{Integral of } F * W \text{ over } (A,B)$ ($W(X) = 1/(X-C)$, ($C.NE.A, C.NE.B$), hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$
- QAWOE DQAWOE - Calculate an approximation to a given definite integral $I = \text{Integral of } F(X) * W(X) \text{ over } (A,B)$, where $W(X) = \text{COS}(\text{OMEGA} * X)$ or $W(X) = \text{SIN}(\text{OMEGA} * X)$, hopefully satisfying the following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
- QAWSE DQAWSE - The routine calculates an approximation result to a given definite integral $I = \text{Integral of } F * W \text{ over } (A,B)$, (where W shows a singular behaviour at the end points, see parameter INTEGR) hopefully satisfying following claim for accuracy $\text{ABS}(I-\text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$.
- QMOMO DQMOMO - This routine computes modified CHEBSYSHEV moments. The K -th modified CHEBYSHEV moment is defined as the integral over $(-1,1)$ of $W(X) * T(K,X)$, where $T(K,X)$ is the CHEBYSHEV POLYNOMIAL of degree K .
- QC25C DQC25C - To compute $I = \text{Integral of } F * W \text{ over } (A,B)$ with error estimate, where $W(X) = 1/(X-C)$
- QC25F DQC25F - To compute the integral $I = \text{Integral of } F(X) \text{ over } (A,B)$ Where $W(X) = \text{COS}(\text{OMEGA} * X)$ or $W(X) = \text{SIN}(\text{OMEGA} * X)$ and to compute $J = \text{Integral of } \text{ABS}(F) \text{ over } (A,B)$. For small value of OMEGA or small intervals (A,B) the 15-point GAUSS-KRONRO rule is used. Otherwise a generalized CLENSHAW-CURTIS method is used.
- QC25S DQC25S - To compute $I = \text{Integral of } F * W \text{ over } (BL, BR)$, with error estimate, where the weight function W has a singular behaviour of ALGEBRAICO-LOGARITHMIC type at the points A and/or B . (BL, BR) is a part of (A, B) .
- QK15W DQK15W - To compute $I = \text{Integral of } F * W \text{ over } (A, B)$, with error estimate $J = \text{Integral of } \text{ABS}(F * W) \text{ over } (A, B)$

- QAGIE DQAGIE - The routine calculates an approximation result to a given integral $I = \text{Integral of } F \text{ over } (\text{BOUND}, +\text{INFINITY})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, \text{BOUND})$ or $I = \text{Integral of } F \text{ over } (-\text{INFINITY}, +\text{INFINITY})$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{MAX}(\text{EPSABS}, \text{EPSREL} * \text{ABS}(I))$
- QAWFE DQAWFE - The routine calculates an approximation result to a given Fourier integral $I = \text{Integral of } F(X) * W(X) \text{ over } (A, \text{INFINITY})$ where $W(X) = \text{COS}(\text{OMEGA} * X)$ or $W(X) = \text{SIN}(\text{OMEGA} * X)$, hopefully satisfying following claim for accuracy $\text{ABS}(I - \text{RESULT}) \leq \text{EPSABS}$.
- QK15I DQK15I - The original (infinite integration range is mapped onto the interval (0,1) and (A,B) is a part of (0,1). it is the purpose to compute $I = \text{Integral of transformed integrand over } (A, B)$, $J = \text{Integral of ABS(Transformed Integrand) over } (A, B)$.

Category I1 Ordinary Differential Equations

single double complx

- BVSUP DBVSUP - Solve a linear two-point boundary value problem using superposition coupled with an orthonormalization procedure and a variable-step integration scheme.
- DERKF DDERKF - Solve initial value problems in ordinary differential equations by the Runge-Kutta-Fehlberg fifth order method. Uses a variable step.
- DEABM DDEABM - Solve initial value problems in ordinary differential equations by an Adams method. Uses both variable (1-12) order and variable step.
- DEBDF DDEBDF - Solve stiff initial value problems in ordinary differential equations using backward differentiation formula. Uses both variable (1-5) order and variable step.
- SDASSL DDASSL - Solve a system of differential equations of the form $G(T, Y, YPRIME) = 0$.
- SDRIV1 DDRIV1 CDRIV1 Solve initial value problems in ordinary differential equations by Gear's method and stiff solver.
- SDRIV2 DDRIV2 CDRIV2 Same as SDRIV1 except allows nonstiff solver and root finding.
- SDRIV3 DDRIV3 CDRIV3 Same as SDRIV1 except handles implicit equations and sophisticated matrix algebra.
- SINTRP DINTP - Approximate the solution at XOUT by evaluating the polynomial computed in STEPS or DSTEPS.

STEPS DSTEPS - Integrate a system of first order ODEs one step.

Category I2 Partial Differential Equations

single double complx

BLKTRI	-	CBLKTR	Solve a block tridiagonal system of linear equations (usually resulting from the discretization of separable two-dimensional elliptic equations).
-	-	CMGNBN	Solve a complex block tridiagonal linear system of equations by a cyclic reduction algorithm .
GENBUN	-	-	Solve using a cyclic reduction algorithm the linear system that results from a finite difference approximation to certain elliptic PDE's on a centered grid.
HSTCRT	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in Cartesian coordinates.
HSTCSP	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
HSTCYL	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in cylindrical coordinates.
HSTPLR	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in polar coordinates.
HSTSSP	-	-	Solve the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in spherical coordinates and on the surface of the unit sphere (radius of 1).
HW3CRT	-	-	Solve the standard seven-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
HWSCRT	-	-	Solve the standard five-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
HWSCSP	-	-	Solve a finite difference approximation to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
HWSCYL	-	-	Solve a standard finite difference approximation to the Helmholtz equation in cylindrical coordinates.
HWSPLR	-	-	Solve a finite difference approximation to the Helmholtz equation in polar coordinates.

HWSSSP	-	-	Solve a finite difference approximation to the Helmholtz equation in spherical coordinates and on the surface of the unit sphere (radius of 1).
POIS3D	-	-	Solve three-dimensional block tridiagonal linear systems arising from finite difference approximations to three-dimensional Poisson equations using the Fourier transform package written by Paul Swarztrauber.
POISTG	-	-	Solve a block tridiagonal system of linear equations that results from a staggered grid finite difference approximation to 2-d elliptic PDE's.
SEPELI	-	-	Automatically discretizes and solves second and fourth (optionally) order finite difference approximations on a uniform grid to the general separable elliptic PDE on a rectangle with any combination of periodic or mixed boundary conditions.
SEPX4	-	-	Solve for either the second or fourth order finite difference approximation to the solution of a separable elliptic equation on a rectangle. Any combination of periodic or mixed boundary conditions is allowed. ***SEPX4 is 3 times faster than SEPELI***

Category J1 Fast Fourier Transform

single double complx

COSQB	-	-	Inverse cosine transform with odd wave numbers. The unnormalized inverse of COSQF.
COSQF	-	-	Forward cosine transform with odd wave numbers.
COSQI	-	-	Initialize for COSQF and COSQB.
COST	-	-	Cosine transform of a real, even sequence.
COSTI	-	-	Initialize for COST.
EZFFTB	-	-	Simplified real, periodic, inverse(backward) transform.
EZFFTF	-	-	Simplified real, periodic, forward transform.
EZFFTI	-	-	Initialize for EZFFTF and EZFFTB.
RFFTB1	-	CFFTB1	Compute the backward fast Fourier transform of a real coefficient array.
RFFTF1	-	CFFTF1	Compute the forward transform of a real, periodic sequence.
RFFTI1	-	CFFTI1	Initialize a work array for RFFTB1 or RFFTF1. periodic sequence.
SINQB	-	-	Inverse sine transform with odd wave numbers.

The unnormalized inverse of SINQF.

SINQF	-	-	Forward sine transform with odd wave numbers.
SINQI	-	-	Initialize for SINQF and SINQB.
SINT	-	-	Sine transform of a real, odd sequence.
SINTI	-	-	Initialize for SINT.

Category K Approximation

single double complx

EFC	DEFC	-	Fit a piece-wise polynomial curve to discrete data. The piece-wise polynomials are represented as B-splines. The fitting is done in a weighted least squares sense.
FC	DFC	-	Fit a piece-wise polynomial curve to discrete data. The piece-wise polynomials are represented as B-splines. The fitting is done in a least squares sense. Equality and inequality constraints can be imposed on the fitted curve.
PCOEF	DPCOEF	-	Convert the POLFIT coefficients to Taylor series form.
POLFIT	DPOLFT	-	Fit a least squares polynomial to discrete data in one variable.
PVALUE	DP1VLU	-	Use the coefficients generated by POLFIT to evaluate the polynomial fit of degree L, along with the first NDER of its derivatives, at a specified point.
SBOCLS	DBOCLS	-	Solve the bounded and constrained least squares problem $EX = F$ with linear constraints $CX = Y$ and bounds on selected values of X and/or Y.
SBOLS	DBOLS	-	Solve the bounded least squares problem $EX = F$ with bounds on selected values of X.
SCOV	DCOV	-	Calculate the covariance matrix for a nonlinear data fitting problem. It is intended to be used after a successful return from either SNLS1 or SNLS1E.
SNLS1	DNLS1	-	Minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. This code is a combination of the MINPACK codes LMDER, LMDIF, and LMSTR.
SNLS1E	DNLS1E	-	The easy-to-use version of SNLS1. This code is a combination of the MINPACK codes LMDER1, LMDIF1, and LMSTR.
LSEI	DLSEI	-	Solve a linearly constrained least squares problem with equality and inequality constraints, and optionally compute a covariance matrix.

WNNLS DWNLS - Solve a linearly constrained least squares problem with equality constraints and nonnegativity constraints on selected variables.

Category L Statistics and Probability

single double complx

CV DCV - Evaluate variance function of curve obtained from routine FC.

RAND - - Generate a uniformly distributed random number.

RGAUSS - - Generate a normally distributed (Gaussian) random number

RUNIF - - A portable uniform random number generator.

Category N Data Handling (I/O, Sorting)

single double complx integr

SBHIN DBHIN - - Read in sparse linear system in Boeing/Harwell format.

SCPPLT DCPPLT - - Plot a SLAP Column format matrix in printer-plot graphical format.

SPPERM DPPERM - IPPERM (also HPPERM) Rearrange a given data vector (X, DX, IX, or HX) according to a prescribed permutation vector.

SPSORT DPSORT - IPSORT (also HPSORT) Return the permutation vector generated by sorting the array X, DX, IX, or HX.

SSORT DSORT - ISORT Sort an array X and optionally make the same interchanges in array Y. A slightly modified QUICKSORT algorithm is used to sort the array in either increasing or decreasing order.

STIN DTIN - Read in SLAP Triad format linear system.

STOUT DTOUT - Write out SLAP Triad format linear system.

Category R1 Machine Constants

single double complx integr

R1MACH D1MACH - Return real (double) machine dependent constants.
- - - I1MACH
Return integer machine dependent constants.

Category R3 Diagnostics and Error Handling

FDUMP Symbolic dump (should be locally written).
NUMXER Return the most recent error number.
XERCLR Reset the current error number to zero.
XERDMP Print the error tables and then clears them.
XERMAX Set maximum number of times any error message is to be printed.
XERMSG Conveniently processes error messages for SLATEC and other libraries.
XGETF Return the current value of error control flag.
XGETUA Return the unit number(s) to which error messages are being sent.
XGETUN Return the (first) output file to which messages are being sent.
XSETF Set the error control flag.
XSETUA Set up to 5 unit numbers to which messages are to be sent.
XSETUN Set the output file to which error messages are to be sent.

Category Z Other (Documentation)

AAAAAA The SLATEC credit and disclaimer notice.
BSPDOC Documentation for the B-spline routines.
EISDOC Documentation for the EISPACK eigenvalue package.
FFTDOC Documentation for the Fast Fourier Transform package.
PCHDOC Documentation for the PCHIP routines. PCHIP is a Fortran package for piecewise cubic Hermite interpolation of data. It features software to produce a monotone and "visually pleasing" interpolant to monotone data.
QPDOC Documentation for the QUADPACK numerical quadrature package.

LINPACK Routines

Cholesky Operations

single double complx

SCHDC	DCHDC	CCHDC	Compute the Cholesky decomposition of a positive definite matrix. A pivoting option allows the user to estimate the condition or rank of the matrix.
SCHDD	DCHDD	CCHDD	Downdate an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.
SCHEX	DCHEX	CCHEX	Update the Cholesky factorization $A=CTRANS(R)*R$ of a positive definite matrix A under diagonal permutations of the form $U*R*E=RR$, where E is a permutation matrix.
SCHUD	DCHUD	CCHUD	Update an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.

General Band Matrices

single double complx

SGBCO	DGBCO	CGBCO	Factor (LU) a band matrix by Gaussian elimination and estimate the condition of the matrix.
SGBDI	DGBDI	CGBDI	Compute the determinant of a band matrix. Use N times for the inverse.
SGBFA	DGBFA	CGBFA	Factor (LU) a band matrix by elimination.
SGBSL	DGBSL	CGBSL	Solve the band system $A*X=B$ or $CTRANS(A)*X=B$.

General Matrices

single double complx

SGECO	DGECO	CGECO	Factor (LU) a matrix by Gaussian elimination and estimate the condition number.
SGEDI	DGEDI	CGEDI	Compute the determinant and inverse of a matrix.
SGEFA	DGEFA	CGEFA	Factor (LU) a matrix by Gaussian elimination.
SGESL	DGESL	CGESL	Solve the system $A*X=B$ or $TRANS(A)*X=B$ using the factors computed by *GECO or *GEFA.

General Tridiagonal Matrices

single double complx

SGTSL DGTSL CGTSL Solve the system $T^*X=B$ where T is a tridiagonal matrix.

Hermitian Positive Definite Band Matrices

single double complx

SPBCO DPBCO CPBCO Factor (LU) a Hermitian positive definite matrix stored in band form and estimate the condition of the matrix.

SPBDI DPBDI CPBDI Compute the determinant of a Hermitian positive definite band matrix using factors from *PBCO or *PBFA. For the inverse use *PBSL.

SPBFA DPBFA CPBFA Factor (LU) a Hermitian positive definite matrix stored in band form.

SPBSL DPBSL CPBSL Solve the Hermitian positive definite band system $A^*X=B$ using factors from *PBCO or *PBFA.

Hermitian Positive Definite Matrices

single double complx

SPOCO DPOCO CPOCO Factor (LU) a Hermitian positive definite matrix and estimate the condition of the matrix.

SPODI DPODI CPODI Compute the determinant and inverse of a Hermitian positive definite matrix using factors of *POCO, *POFA or *QRDC.

SPOFA DPOFA CPOFA Factor (LU) a Hermitian positive definite matrix.

SPOSL DPOSL CPOSL Solve the Hermitian positive definite system $A^*X=B$ using the factors computed by *POCO or *POFA.

(PACKED FORM)

SPPCO DPPCO CPPCO Factor (LU) a Hermitian positive definite matrix stored in packed form and estimate the condition of the matrix.

SPPDI DPPDI CPPDI Compute the determinant and inverse of a Hermitian positive definite matrix using factors from *PPCO or *PPFA

SPPFA DPPFA CPPFA Factor (LU) a Hermitian positive definite matrix stored in packed form.

SPPSL DPPSL CPPSL Solve the Hermitian positive definite system $A^*X=B$ using the factors computed by *PPCO or *PPFA.

Positive Definite Tridiagonal Matrices

single double complx

SPTSL DPTSL CPTSL Solve a positive definite tridiagonal system.

Symmetric Matrices

single double complx

SSICO DSICO CSICO Factor (LU) a symmetric matrix by elimination with symmetric pivoting and estimate the condition number.

SSIDI DSIDI CSIDI Compute the determinant and inverse of a symmetric matrix using the factors from *SIFA.

SSIFA DSIFA CSIFA Factor (LU) a symmetric matrix by elimination with symmetric pivoting.

SSISL DSISL CSISL Solve the symmetric system $A*X=B$ using factors from *SIFA.

(PACKED FORM)

single double complx

SSPCO DSPCO CSPCO Factor (LU) a symmetric matrix stored in packed form by elimination with symmetric pivoting and estimate the condition of the matrix.

SSPDI DSPDI CSPDI Compute the determinant and inverse of a symmetric matrix stored in packed form using factors from *SPFA.

SSPFA DSPFA CSPFA Factor (LU) a symmetric matrix stored in packed form by elimination with symmetric pivoting.

SSPSL DSPSL CSPSL Solve the symmetric system $A*X=B$ using factors from *SPFA.

Triangular Matrices

single double complx

STRCO DTRCO CTRCO Estimates the condition of a triangular system.

STRDI DTRDI CTRDI Compute the determinant and inverse of a triangular matrix.

STRSL DTRSL CTRSL Solve systems of the form $T*X=B$ or $TRANS(T)*X=B$ where T is a triangular matrix.

Complex Hermitian Matrices

single	double	complx	
-----	-----	-----	
-	-	CHICO	Factor (LU) a complex Hermitian matrix by elimination with symmetric pivoting and estimate the condition number.
-	-	CHIDI	Compute the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHIFA.
-	-	CHIFA	Factor (LU) a Hermitian matrix by elimination with symmetric pivoting.
-	-	CHISL	Solve the Hermitian system $A*X=B$ using factors of CHIFA. (PACKED FORM)
-	-	CHPCO	Factor (LU) a complex Hermitian matrix (packed form) by elimination with symmetric pivoting and estimate the condition of the matrix.
-	-	CHPDI	Compute the determinant, inertia and inverse of a complex Hermitian matrix (packed form) using the factors from CHPFA.
-	-	CHPFA	Factor (LU) a complex Hermitian matrix (packed form) by elimination with symmetric pivoting.
-	-	CHPSL	Solve the Hermitian system $A*X=B$ using factors of CHPFA.

EISPACK Routines

single	double	complx	
-----	-----	-----	
RS	-	CH	Computes eigenvalues and, optionally, eigenvectors of real symmetric (complex Hermitian) matrix.
RSP	-	-	Compute eigenvalues and, optionally, eigenvectors of real symmetric matrix packed into a one dimensional array.
RG	-	CG	Computes eigenvalues and, optionally, eigenvectors of a real (complex) general matrix.
BISECT	-	-	Compute eigenvalues of symmetric tridiagonal matrix given interval using Sturm sequencing.
IMTQL1	-	-	Computes eigenvalues of symmetric tridiagonal matrix implicit QL method.
IMTQL2	-	-	Computes eigenvalues and eigenvectors of symmetric tridiagonal matrix using implicit QL method.

IMTQLV	-	-	Computes eigenvalues of symmetric tridiagonal matrix by the implicit QL method. Eigenvectors may be computed later.
RATQR	-	-	Computes largest or smallest eigenvalues of symmetric tridiagonal matrix using rational QR method with Newton correction.
RST	-	-	Compute eigenvalues and, optionally, eigenvectors of real symmetric tridiagonal matrix.
RT	-	-	Compute eigenvalues and eigenvectors of a special real tridiagonal matrix.
TQL1	-	-	Compute eigenvalues of symmetric tridiagonal matrix by QL method.
TQL2	-	-	Compute eigenvalues and eigenvectors of symmetric tridiagonal matrix.
TQLRAT	-	-	Computes eigenvalues of symmetric tridiagonal matrix a rational variant of the QL method.
TRIDIB	-	-	Computes eigenvalues of symmetric tridiagonal matrix given interval using Sturm sequencing.
TSTURM	-	-	Computes eigenvalues of symmetric tridiagonal matrix given interval and eigenvectors by Sturm sequencing. This subroutine is a translation of the ALGOL procedure TRISTURM by Peters and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 418-439(1971).
BQR	-	-	Computes some of the eigenvalues of a real symmetric matrix using the QR method with shifts of origin.
RSB	-	-	Computes eigenvalues and, optionally, eigenvectors of symmetric band matrix
RSG	-	-	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $A*X=(LAMBDA)*B*X$
RSGAB	-	-	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $A*B*X=(LAMBDA)*X$
RSGBA	-	-	Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $B*A*X=(LAMBDA)*X$
RGG	-	-	Computes eigenvalues and eigenvectors for real generalized eigenproblem: $A*X=(LAMBDA)*B*X$.
BALANC	-	CBAL	Balances a general real (complex) matrix and isolates eigenvalues whenever possible.
BANDR	-	-	Reduces real symmetric band matrix to symmetric tridiagonal matrix and, optionally, accumulates orthogonal similarity transformations.
HTRID3	-	-	Reduces complex Hermitian (packed) matrix to real symmetric tridiagonal matrix by unitary similarity transformations.

HTRIDI	-	-	Reduces complex Hermitian matrix to real symmetric tridiagonal matrix using unitary similarity transformations.
TRED1	-	-	Reduce real symmetric matrix to symmetric tridiagonal matrix using orthogonal similarity transformations.
TRED2	-	-	Reduce real symmetric matrix to symmetric tridiagonal matrix using and accumulating orthogonal transformations.
TRED3	-	-	Reduce real symmetric matrix stored in packed form to symmetric tridiagonal matrix using orthogonal transformations.
ELMHES	-	COMHES	Reduces real (complex) general matrix to upper Hessenberg form using stabilized elementary similarity transformations.
ORTHES	-	CORTH	Reduces real (complex) general matrix to upper Hessenberg form orthogonal (unitary) similarity transformations.
QZHES	-	-	The first step of the QZ algorithm for solving generalized matrix eigenproblems. Accepts a pair of real general matrices and reduces one of them to upper Hessenberg and the other to upper triangular form using orthogonal transformations. Usually followed by QZIT, QZVAL, QZ
QZIT	-	-	The second step of the QZ algorithm for generalized eigenproblems. Accepts an upper Hessenberg and an upper triangular matrix and reduces the former to quasi-triangular form while preserving the form of the latter. Usually preceded by QZHES and followed by QZVAL and QZVEC.
FIGI	-	-	Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.
FIGI2	-	-	Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.
REDUC	-	-	Reduces generalized symmetric eigenproblem $A*X=(LAMBDA)*B*X$, to standard symmetric eigenproblem using Cholesky factorization.
REDUC2	-	-	Reduces certain generalized symmetric eigenproblems standard symmetric eigenproblem, using Cholesky factorization.
-	-	COMLR	Computes eigenvalues of a complex upper Hessenberg m using the modified LR method.
-	-	COMLR2	Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix using modified LR method.
HQR	-	COMQR	Computes eigenvalues of a real (complex) upper Hessenberg matrix using the QR method.

HQR2	-	COMQR2	Computes eigenvalues and eigenvectors of real (complex) upper Hessenberg matrix using QR method.
INVIT	-	CINVIT	Computes eigenvectors of real (complex) Hessenberg matrix associated with specified eigenvalues by inverse iteration.
QZVAL	-	-	The third step of the QZ algorithm for generalized eigenproblems. Accepts a pair of real matrices, one quasi-triangular form and the other in upper triangular form and computes the eigenvalues of the associated eigenproblem. Usually preceded by QZHES, QZIT, and followed by QZVEC.
BANDV	-	-	Forms eigenvectors of real symmetric band matrix associated with a set of ordered approximate eigenvalue by inverse iteration.
QZVEC	-	-	The optional fourth step of the QZ algorithm for generalized eigenproblems. Accepts a matrix in quasi-triangular form and another in upper triangular and computes the eigenvectors of the triangular problem and transforms them back to the original coordinates. Usually preceded by QZHES, QZIT, QZVAL.
TINVIT	-	-	Eigenvectors of symmetric tridiagonal matrix corresponding to some specified eigenvalues, using inverse iteration.
BAKVEC	-	-	Forms eigenvectors of certain real non-symmetric tridiagonal matrix from symmetric tridiagonal matrix output from FIGI.
BALBAK	-	CBABK2	Forms eigenvectors of real (complex) general matrix from eigenvectors of matrix output from BALANC (CBAL).
ELMBAK	-	COMBAK	Forms eigenvectors of real (complex) general matrix from eigenvectors of upper Hessenberg matrix output from ELMHES (COMHES).
ELTRAN	-	-	Accumulates the stabilized elementary similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ELMHES.
HTRIB3	-	-	Computes eigenvectors of complex Hermitian matrix from eigenvectors of real symmetric tridiagonal matrix output from HTRID3.
HTRIBK	-	-	Forms eigenvectors of complex Hermitian matrix from eigenvectors of real symmetric tridiagonal matrix output from HTRIDI.
ORTBAK	-	CORTB	Forms eigenvectors of general real (complex) matrix from eigenvectors of upper Hessenberg matrix output from ORTHES (CORTH).
ORTTRAN	-	-	Accumulates orthogonal similarity transformations in reduction of real general matrix by ORTHES.

REBAK	-	-	Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC REDUC2.
REBAKB	-	-	Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC2
TRBAK1	-	-	Forms the eigenvectors of real symmetric matrix from eigenvectors of symmetric tridiagonal matrix formed by TRED1.
TRBAK3	-	-	Forms eigenvectors of real symmetric matrix from the eigenvectors of symmetric tridiagonal matrix formed by TRED3.
MINFIT	-	-	Compute singular value decomposition of rectangular matrix and solve related linear least squares problem.

SLATEC Subroutine Dictionary

This is an alphabetical list of SLATEC routine names that indicates the GAMS category and the general function of each routine. To see a list of the GAMS subject categories used to group the SLATEC subroutines by function performed, consult the "Subject Category Overview" section above (keyword: [categories](#) (page 39)). To see an annotated (descriptive) list of the subroutines in any specific category, consult the section called "SLATEC Routines by Subject Category" (keyword: [routines](#) (page 41)) or use the category code (such as d6 or h2) as a link or keyword. To see detailed documentation for a specific subroutine, consult the alphabetically arranged catalog writeups SLATEC2 through SLATEC5 (where each subroutine's name is a keyword to facilitate viewing those documents by section online).

Routine Name	Gams Cat.	Function Performed (keywords or links)
AAAAAA	z	documentation
ACOSH	c	elementary-functions, special-functions
AI	c	elementary-functions, special-functions
AIE	c	elementary-functions, special-functions
ALBETA	c	elementary-functions, special-functions
ALGAMS	c	elementary-functions, special-functions
ALI	c	elementary-functions, special-functions
ALNGAM	c	elementary-functions, special-functions
ALNREL	c	elementary-functions, special-functions
ASINH	c	elementary-functions, special-functions
ATANH	c	elementary-functions, special-functions
AVINT	h2	quadrature, definite-integrals
BAKVEC	eispack	
BALANC	eispack	
BALBAK	eispack	
BANDR	eispack	
BANDV	eispack	
BESI	c	elementary-functions, special-functions
BESIO	c	elementary-functions, special-functions
BESIOE	c	elementary-functions, special-functions
BESI1	c	elementary-functions, special-functions
BESI1E	c	elementary-functions, special-functions
BESJ	c	elementary-functions, special-functions
BESJ0	c	elementary-functions, special-functions
BESJ1	c	elementary-functions, special-functions
BESK	c	elementary-functions, special-functions
BESK0	c	elementary-functions, special-functions
BESK0E	c	elementary-functions, special-functions
BESK1	c	elementary-functions, special-functions
BESK1E	c	elementary-functions, special-functions
BESKES	c	elementary-functions, special-functions
BESKS	c	elementary-functions, special-functions
BESY	c	elementary-functions, special-functions
BESY0	c	elementary-functions, special-functions
BESY1	c	elementary-functions, special-functions
BETA	c	elementary-functions, special-functions
BETAI	c	elementary-functions, special-functions
BFQAD	e	interpolation
BI	c	elementary-functions, special-functions

BIE	<u>c</u>	elementary-functions, special-functions
BINOM	<u>c</u>	elementary-functions, special-functions
BINT4	<u>e</u>	interpolation
BINTK	<u>e</u>	interpolation
BISECT	<u>eispack</u>	
BLKTRI	<u>i2</u>	partial-differential-equations
BNDACC	<u>d9</u>	overdetermined-systems, least-squares
BNDSOL	<u>d9</u>	overdetermined-systems, least-squares
BQR	<u>eispack</u>	
BSKIN	<u>c</u>	elementary-functions, special-functions
BSPDOC	<u>z</u>	documentation
BSPDR	<u>e</u>	interpolation
BSPEV	<u>e</u>	interpolation
BSPPP	<u>e</u>	interpolation
BSPVD	<u>e</u>	interpolation
BSPVN	<u>e</u>	interpolation
BSQAD	<u>e</u>	interpolation
BVALU	<u>e</u>	interpolation
BVSUP	<u>i1</u>	ordinary-differential-equations
COLGMC	<u>c</u>	elementary-functions, special-functions
CACOS	<u>c</u>	elementary-functions, special-functions
CACOSH	<u>c</u>	elementary-functions, special-functions
CAIRY	<u>c</u>	elementary-functions, special-functions
CARG	<u>c</u>	elementary-functions, special-functions
CASIN	<u>c</u>	elementary-functions, special-functions
CASINH	<u>c</u>	elementary-functions, special-functions
CATAN	<u>c</u>	elementary-functions, special-functions
CATAN2	<u>c</u>	elementary-functions, special-functions
CATANH	<u>c</u>	elementary-functions, special-functions
CAXPY	<u>d1a</u>	vector-operations
CBABK2	<u>eispack</u>	
CBAL	<u>eispack</u>	
CBESH	<u>c</u>	elementary-functions, special-functions
CBESI	<u>c</u>	elementary-functions, special-functions
CBESJ	<u>c</u>	elementary-functions, special-functions
CBESK	<u>c</u>	elementary-functions, special-functions
CBESY	<u>c</u>	elementary-functions, special-functions
CBETA	<u>c</u>	elementary-functions, special-functions
CBIRY	<u>c</u>	elementary-functions, special-functions
CBLKTR	<u>i2</u>	partial-differential-equations
CBRT	<u>c</u>	elementary-functions, special-functions
CCBRT	<u>c</u>	elementary-functions, special-functions
CCHDC	<u>linpack</u>	cholesky-operations
CCHDD	<u>linpack</u>	cholesky-operations
CCHEX	<u>linpack</u>	cholesky-operations
CCHUD	<u>linpack</u>	cholesky-operations
CCOPY	<u>d1a</u>	vector-operations
CCOSH	<u>c</u>	elementary-functions, special-functions
CCOT	<u>c</u>	elementary-functions, special-functions
CDCDOT	<u>d1a</u>	vector-operations
CDOTC	<u>d1a</u>	vector-operations
CDOTU	<u>d1a</u>	vector-operations
CDRIV1	<u>i1</u>	ordinary-differential-equations
CDRIV2	<u>i1</u>	ordinary-differential-equations
CDRIV3	<u>i1</u>	ordinary-differential-equations
CEXPRL	<u>c</u>	elementary-functions, special-functions
CFFTb1	<u>j1</u>	fast-fourier-transforms
CFFTf1	<u>j1</u>	fast-fourier-transforms
CFFTt1	<u>j1</u>	fast-fourier-transforms
CG	<u>eispack</u>	

CGAMMA	<u>c</u>	elementary-functions, special-functions
CGAMR	<u>c</u>	elementary-functions, special-functions
CGBCO	<u>linpack</u>	general-band
CGBDI	<u>linpack</u>	general-band
CGBFA	<u>linpack</u>	general-band
CGBSL	<u>linpack</u>	general-band
CGECO	<u>linpack</u>	general
CGEDI	<u>linpack</u>	general
CGEEV	<u>d4</u>	eigenvalues, eigenvectors
CGEFA	<u>linpack</u>	general
CGEFS	<u>d2</u>	linear-equations
CGEIR	<u>d2</u>	linear-equations
CGEMM	<u>d1b</u>	matrix-operations
CGEMV	<u>d1b</u>	matrix-operations
CGERC	<u>d1b</u>	matrix-operations
CGERU	<u>d1b</u>	matrix-operations
CGESL	<u>linpack</u>	general
CGTSL	<u>linpack</u>	general-tridiagonal
CH	<u>eispack</u>	
CHBMV	<u>d1b</u>	matrix-operations
CHEMM	<u>d1b</u>	matrix-operations
CHEMV	<u>d1b</u>	matrix-operations
CHER	<u>d1b</u>	matrix-operations
CHER2	<u>d1b</u>	matrix-operations
CHER2K	<u>d1b</u>	matrix-operations
CHERK	<u>d1b</u>	matrix-operations
CHFDV	<u>e</u>	interpolation
CHFEV	<u>e</u>	interpolation
CHICO	<u>linpack</u>	complex-hermitian
CHIDI	<u>linpack</u>	complex-hermitian
CHIEV	<u>d4</u>	eigenvalues, eigenvectors
CHIFA	<u>linpack</u>	complex-hermitian
CHISL	<u>linpack</u>	complex-hermitian
CHKDER	<u>f</u>	nonlinear-equations
CHPCO	<u>linpack</u>	complex-hermitian
CHPDI	<u>linpack</u>	complex-hermitian
CHPFA	<u>linpack</u>	complex-hermitian
CHPMV	<u>d1b</u>	matrix-operations
CHPR	<u>d1b</u>	matrix-operations
CHPR2	<u>d1b</u>	matrix-operations
CHPSL	<u>linpack</u>	complex-hermitian
CHU	<u>c</u>	elementary-functions, special-functions
CINVIT	<u>eispack</u>	
CLBETA	<u>c</u>	elementary-functions, special-functions
CLNGAM	<u>c</u>	elementary-functions, special-functions
CLNREL	<u>c</u>	elementary-functions, special-functions
CLOG10	<u>c</u>	elementary-functions, special-functions
CMGNBN	<u>i2</u>	partial-differential-equations
CNBCO	<u>d2</u>	linear-equations
CNBDI	<u>d3</u>	determinants
CNBFA	<u>d2</u>	linear-equations
CNBFS	<u>d2</u>	linear-equations
CNBIR	<u>d2</u>	linear-equations
CNBSL	<u>d2</u>	linear-equations
COMBAK	<u>eispack</u>	
COMHES	<u>eispack</u>	
COMLR	<u>eispack</u>	
COMLR2	<u>eispack</u>	
COMQR	<u>eispack</u>	
COMQR2	<u>eispack</u>	

CORTB	<u>eispack</u>	
CORTH	<u>eispack</u>	
COSDG	<u>c</u>	elementary-functions, special-functions
COSQB	<u>j1</u>	fast-fourier-transforms
COSQF	<u>j1</u>	fast-fourier-transforms
COSQI	<u>j1</u>	fast-fourier-transforms
COST	<u>j1</u>	fast-fourier-transforms
COSTI	<u>j1</u>	fast-fourier-transforms
COT	<u>c</u>	elementary-functions, special-functions
CPBCO	<u>linpack</u>	hermitian-positive-definite-band
CPBDI	<u>linpack</u>	hermitian-positive-definite-band
CPBFA	<u>linpack</u>	hermitian-positive-definite-band
CPBSL	<u>linpack</u>	hermitian-positive-definite-band
CPOCO	<u>linpack</u>	hermitian-positive-definite
CPODI	<u>linpack</u>	hermitian-positive-definite
CPOFA	<u>linpack</u>	hermitian-positive-definite
CPOFS	<u>d2</u>	linear-equations
CPOIR	<u>d2</u>	linear-equations
CPOSL	<u>linpack</u>	hermitian-positive-definite
CPPCO	<u>linpack</u>	hermitian-positive-definite
CPPDI	<u>linpack</u>	hermitian-positive-definite
CPPFA	<u>linpack</u>	hermitian-positive-definite
CPPSL	<u>linpack</u>	hermitian-positive-definite
CPQR79	<u>f</u>	nonlinear-equations
CPSI	<u>c</u>	elementary-functions, special-functions
CPTSL	<u>linpack</u>	positive-definite-tridiagonal
CPZERO	<u>f</u>	nonlinear-equations
CQRDC	<u>d5</u>	qr-decomposition
CQRSL	<u>d5</u>	qr-decomposition
CROTG	<u>d1a</u>	vector-operations
CSCAL	<u>d1a</u>	vector-operations
CSEVL	<u>c</u>	elementary-functions, special-functions
CSICO	<u>linpack</u>	symmetric
CSIDI	<u>linpack</u>	symmetric
CSIFA	<u>linpack</u>	symmetric
CSINH	<u>c</u>	elementary-functions, special-functions
CSISL	<u>linpack</u>	symmetric
CSPCO	<u>linpack</u>	symmetric
CSPDI	<u>linpack</u>	symmetric
CSPFA	<u>linpack</u>	symmetric
CSPSL	<u>linpack</u>	symmetric
CSROT	<u>d1a</u>	vector-operations
CSSCAL	<u>d1a</u>	vector-operations
CSVDC	<u>d6</u>	singular-value-decomposition
CSWAP	<u>d1a</u>	vector-operations
CSYMM	<u>d1b</u>	matrix-operations
CSYR2K	<u>d1b</u>	matrix-operations
CSYRK	<u>d1b</u>	matrix-operations
CTAN	<u>c</u>	elementary-functions, special-functions
CTANH	<u>c</u>	elementary-functions, special-functions
CTBMV	<u>d1b</u>	matrix-operations
CTBSV	<u>d1b</u>	matrix-operations
CTPMV	<u>d1b</u>	matrix-operations
CTPSV	<u>d1b</u>	matrix-operations
CTRCO	<u>linpack</u>	triangular
CTRDI	<u>linpack</u>	triangular
CTRMM	<u>d1b</u>	matrix-operations
CTRMV	<u>d1b</u>	matrix-operations
CTRSL	<u>linpack</u>	triangular
CTRSM	<u>d1b</u>	matrix-operations

CTRSV	<u>d1b</u>	matrix-operations
CV	<u>l</u>	statistics
D1MACH	<u>r1</u>	machine-constants
D9PAK	<u>a</u>	arithmetic-functions
D9UPAK	<u>a</u>	arithmetic-functions
DACOSH	<u>c</u>	elementary-functions, special-functions
DAI	<u>c</u>	elementary-functions, special-functions
DAIE	<u>c</u>	elementary-functions, special-functions
DASINH	<u>c</u>	elementary-functions, special-functions
DASUM	<u>d1a</u>	vector-operations
DATANH	<u>c</u>	elementary-functions, special-functions
DAVINI	<u>h2</u>	quadrature, definite-integrals
DAWS	<u>c</u>	elementary-functions, special-functions
DAXPY	<u>d1a</u>	vector-operations
DBCQ	<u>d2</u>	linear-equations
DBESI	<u>c</u>	elementary-functions, special-functions
DBESIO	<u>c</u>	elementary-functions, special-functions
DBESI1	<u>c</u>	elementary-functions, special-functions
DBESJ	<u>c</u>	elementary-functions, special-functions
DBESJ0	<u>c</u>	elementary-functions, special-functions
DBESJ1	<u>c</u>	elementary-functions, special-functions
DBESK	<u>c</u>	elementary-functions, special-functions
DBESK0	<u>c</u>	elementary-functions, special-functions
DBESK1	<u>c</u>	elementary-functions, special-functions
DBESKS	<u>c</u>	elementary-functions, special-functions
DBESY	<u>c</u>	elementary-functions, special-functions
DBESY0	<u>c</u>	elementary-functions, special-functions
DBESY1	<u>c</u>	elementary-functions, special-functions
DBETA	<u>c</u>	elementary-functions, special-functions
DBETAI	<u>c</u>	elementary-functions, special-functions
DBFQAD	<u>e</u>	interpolation
DBHIN	<u>n</u>	data-handling
DBHIN	<u>n</u>	data-handling
DBI	<u>c</u>	elementary-functions, special-functions
DBIE	<u>c</u>	elementary-functions, special-functions
DBINOM	<u>c</u>	elementary-functions, special-functions
DBINT4	<u>e</u>	interpolation
DBINTK	<u>e</u>	interpolation
DBNDAC	<u>d9</u>	overdetermined-systems, least-squares
DBNDSL	<u>d9</u>	overdetermined-systems, least-squares
DBOCLS	<u>k</u>	approximation
DBOLS	<u>k</u>	approximation
DBSI0E	<u>c</u>	elementary-functions, special-functions
DBSI1E	<u>c</u>	elementary-functions, special-functions
DBSK0E	<u>c</u>	elementary-functions, special-functions
DBSK1E	<u>c</u>	elementary-functions, special-functions
DBSKES	<u>c</u>	elementary-functions, special-functions
DBSKIN	<u>c</u>	elementary-functions, special-functions
DBSPDR	<u>e</u>	interpolation
DBSPEV	<u>e</u>	interpolation
DBSPPP	<u>e</u>	interpolation
DBSPVD	<u>e</u>	interpolation
DBSPVN	<u>e</u>	interpolation
DBSQAD	<u>e</u>	interpolation
DBVALU	<u>e</u>	interpolation
DBVSUP	<u>i1</u>	ordinary-differential-equations
DCBRT	<u>c</u>	elementary-functions, special-functions
DCDOT	<u>d1a</u>	vector-operations
DCG	<u>d2</u>	linear-equations
DCGN	<u>d2</u>	linear-equations

DCGS	<u>d2</u>	linear-equations
DCHDC	<u>linpack</u>	cholesky-operations
DCHDD	<u>linpack</u>	cholesky-operations
DCHEX	<u>linpack</u>	cholesky-operations
DCHFVDV	<u>e</u>	interpolation
DCHFV	<u>e</u>	interpolation
DCHU	<u>c</u>	elementary-functions, special-functions
DCHUD	<u>linpack</u>	cholesky-operations
DCKDER	<u>f</u>	nonlinear-equations
DCOPY	<u>d1a</u>	vector-operations
DCOPYM	<u>d1a</u>	vector-operations
DCOSDG	<u>c</u>	elementary-functions, special-functions
DCOT	<u>c</u>	elementary-functions, special-functions
DCOV	<u>k</u>	approximation
DCPPLT	<u>n</u>	data-handling
DCPPLT	<u>n</u>	data-handling
DCSEVL	<u>c</u>	elementary-functions, special-functions
DCV	<u>l</u>	statistics
DDASSL	<u>i1</u>	ordinary-differential-equations
DDAWS	<u>c</u>	elementary-functions, special-functions
DDEABM	<u>i1</u>	ordinary-differential-equations
DDEBDF	<u>i1</u>	ordinary-differential-equations
DDERKF	<u>i1</u>	ordinary-differential-equations
DDOT	<u>d1a</u>	vector-operations
DDRIV1	<u>i1</u>	ordinary-differential-equations
DDRIV2	<u>i1</u>	ordinary-differential-equations
DDRIV3	<u>i1</u>	ordinary-differential-equations
DE1	<u>c</u>	elementary-functions, special-functions
DEABM	<u>i1</u>	ordinary-differential-equations
DEBDF	<u>i1</u>	ordinary-differential-equations
DEFC	<u>k</u>	approximation
DEI	<u>c</u>	elementary-functions, special-functions
DERF	<u>c</u>	elementary-functions, special-functions
DERFC	<u>c</u>	elementary-functions, special-functions
DERKF	<u>i1</u>	ordinary-differential-equations
DEXINT	<u>c</u>	elementary-functions, special-functions
DEXPRL	<u>c</u>	elementary-functions, special-functions
DFAC	<u>c</u>	elementary-functions, special-functions
DFC	<u>k</u>	approximation
DFZERO	<u>f</u>	nonlinear-equations
DGAMI	<u>c</u>	elementary-functions, special-functions
DGAMIC	<u>c</u>	elementary-functions, special-functions
DGAMIT	<u>c</u>	elementary-functions, special-functions
DGAMLM	<u>c</u>	elementary-functions, special-functions
DGAMMA	<u>c</u>	elementary-functions, special-functions
DGAMR	<u>c</u>	elementary-functions, special-functions
DGAUS8	<u>h2</u>	quadrature, definite-integrals
DGBCO	<u>linpack</u>	general-band
DGBDI	<u>linpack</u>	general-band
DGBFA	<u>linpack</u>	general-band
DGBSL	<u>linpack</u>	general-band
DGECO	<u>linpack</u>	general
DGEDI	<u>linpack</u>	general
DGEFA	<u>linpack</u>	general
DGEFS	<u>d2</u>	linear-equations
DGEMM	<u>d1b</u>	matrix-operations
DGEMV	<u>d1b</u>	matrix-operations
DGER	<u>d1b</u>	matrix-operations
DGESL	<u>linpack</u>	general
DGLSS	<u>d9</u>	overdetermined-systems, least-squares

DGMRES	<u>d2</u>	linear-equations
DGTSL	<u>linpack</u>	general-tridiagonal
DHFTI	<u>d9</u>	overdetermined-systems, least-squares
DINTP	<u>i1</u>	ordinary-differential-equations
DINTRV	<u>e</u>	interpolation
DIR	<u>d2</u>	linear-equations
DLBETA	<u>c</u>	elementary-functions, special-functions
DLGAMS	<u>c</u>	elementary-functions, special-functions
DLI	<u>c</u>	elementary-functions, special-functions
DLLSIA	<u>d9</u>	overdetermined-systems, least-squares
DLITI2	<u>d2</u>	linear-equations
DLNGAM	<u>c</u>	elementary-functions, special-functions
DLNREL	<u>c</u>	elementary-functions, special-functions
DLPDOG	<u>d2</u>	linear-equations
DLSEI	<u>k</u>	approximation
DNBCO	<u>d2</u>	linear-equations
DNBDI	<u>d3</u>	determinants
DNBFA	<u>d2</u>	linear-equations
DNBFS	<u>d2</u>	linear-equations
DNBSL	<u>d2</u>	linear-equations
DNLS1	<u>k</u>	approximation
DNLS1E	<u>k</u>	approximation
DNRM2	<u>d1a</u>	vector-operations
DNSQ	<u>f</u>	nonlinear-equations
DNSQE	<u>f</u>	nonlinear-equations
DOMN	<u>d2</u>	linear-equations
DP1VLU	<u>k</u>	approximation
DPBCO	<u>linpack</u>	hermitian-positive-definite-band
DPBDI	<u>linpack</u>	hermitian-positive-definite-band
DPBFA	<u>linpack</u>	hermitian-positive-definite-band
DPBSL	<u>linpack</u>	hermitian-positive-definite-band
DPCHBS	<u>e</u>	interpolation
DPCHCM	<u>e</u>	interpolation
DPCHFD	<u>e</u>	interpolation
DPCHFE	<u>e</u>	interpolation
DPCHIA	<u>e</u>	interpolation
DPCHIC	<u>e</u>	interpolation
DPCHID	<u>e</u>	interpolation
DPCHIM	<u>e</u>	interpolation
DPCHSP	<u>e</u>	interpolation
DPCOEF	<u>k</u>	approximation
DPFQAD	<u>e</u>	interpolation
DPLINT	<u>e</u>	interpolation
DPOCH	<u>c</u>	elementary-functions, special-functions
DPOCH1	<u>c</u>	elementary-functions, special-functions
DPOCO	<u>linpack</u>	hermitian-positive-definite
DPODI	<u>linpack</u>	hermitian-positive-definite
DPOFA	<u>linpack</u>	hermitian-positive-definite
DPOFS	<u>d2</u>	linear-equations
DPOLCF	<u>e</u>	interpolation
DPOLFT	<u>k</u>	approximation
DPOLVL	<u>e</u>	interpolation
DPOSL	<u>linpack</u>	hermitian-positive-definite
DPPCO	<u>linpack</u>	hermitian-positive-definite
DPPDI	<u>linpack</u>	hermitian-positive-definite
DPPERM	<u>n</u>	data-handling
DPPFA	<u>linpack</u>	hermitian-positive-definite
DPPQAD	<u>e</u>	interpolation
DPPSL	<u>linpack</u>	hermitian-positive-definite
DPPVAL	<u>e</u>	interpolation

DPSI	<u>c</u>	elementary-functions, special-functions
DPSIFN	<u>c</u>	elementary-functions, special-functions
DPSORT	<u>n</u>	data-handling
DPTSL	<u>linpack</u>	positive-definite-tridiagonal
DQAG	<u>h2</u>	quadrature, definite-integrals
DQAGE	<u>h2</u>	quadrature, definite-integrals
DQAGI	<u>h2</u>	quadrature, definite-integrals
DQAGIE	<u>h2</u>	quadrature, definite-integrals
DQAGP	<u>h2</u>	quadrature, definite-integrals
DQAGPE	<u>h2</u>	quadrature, definite-integrals
DQAGS	<u>h2</u>	quadrature, definite-integrals
DQAGSE	<u>h2</u>	quadrature, definite-integrals
DQAWC	<u>h2</u>	quadrature, definite-integrals
DQAWCE	<u>h2</u>	quadrature, definite-integrals
DQAWF	<u>h2</u>	quadrature, definite-integrals
DQAWFE	<u>h2</u>	quadrature, definite-integrals
DQAWO	<u>h2</u>	quadrature, definite-integrals
DQAWOE	<u>h2</u>	quadrature, definite-integrals
DQAWS	<u>h2</u>	quadrature, definite-integrals
DQAWSE	<u>h2</u>	quadrature, definite-integrals
DQC25C	<u>h2</u>	quadrature, definite-integrals
DQC25F	<u>h2</u>	quadrature, definite-integrals
DQC25S	<u>h2</u>	quadrature, definite-integrals
DQDOTA	<u>d1a</u>	vector-operations
DQDOTI	<u>d1a</u>	vector-operations
DQK15	<u>h2</u>	quadrature, definite-integrals
DQK15I	<u>h2</u>	quadrature, definite-integrals
DQK15W	<u>h2</u>	quadrature, definite-integrals
DQK21	<u>h2</u>	quadrature, definite-integrals
DQK31	<u>h2</u>	quadrature, definite-integrals
DQK41	<u>h2</u>	quadrature, definite-integrals
DQK51	<u>h2</u>	quadrature, definite-integrals
DQK61	<u>h2</u>	quadrature, definite-integrals
DQMOMO	<u>h2</u>	quadrature, definite-integrals
DQNC79	<u>h2</u>	quadrature, definite-integrals
DQNG	<u>h2</u>	quadrature, definite-integrals
DQRDC	<u>d5</u>	qr-decomposition
DQRSL	<u>d5</u>	qr-decomposition
DRC	<u>c</u>	elementary-functions, special-functions
DRC3JJ	<u>c</u>	elementary-functions, special-functions
DRC3JM	<u>c</u>	elementary-functions, special-functions
DRC6J	<u>c</u>	elementary-functions, special-functions
DRD	<u>c</u>	elementary-functions, special-functions
DRF	<u>c</u>	elementary-functions, special-functions
DRJ	<u>c</u>	elementary-functions, special-functions
DROT	<u>d1a</u>	vector-operations
DROTG	<u>d1a</u>	vector-operations
DROTM	<u>d1a</u>	vector-operations
DROTMG	<u>d1a</u>	vector-operations
DS2LT	<u>d2</u>	linear-equations
DS2Y	<u>d1b</u>	matrix-operations
DSBMV	<u>d1b</u>	matrix-operations
DSCAL	<u>d1a</u>	vector-operations
DSD2S	<u>d2</u>	linear-equations
DSBCG	<u>d2</u>	linear-equations
DSDCG	<u>d2</u>	linear-equations
DSDCGN	<u>d2</u>	linear-equations
DSDCGS	<u>d2</u>	linear-equations
DSDGMR	<u>d2</u>	linear-equations
DSDI	<u>d1b</u>	matrix-operations

DSDOMN	<u>d2</u>	linear-equations
DSDOT	<u>d1a</u>	vector-operations
DSDS	<u>d2</u>	linear-equations
DSDSCL	<u>d2</u>	linear-equations
DSGS	<u>d2</u>	linear-equations
DSICGC	<u>d2</u>	linear-equations
DSICO	<u>linpack</u>	symmetric
DSICS	<u>d2</u>	linear-equations
DSIDI	<u>linpack</u>	symmetric
DSIFA	<u>linpack</u>	symmetric
DSILUR	<u>d2</u>	linear-equations
DSILUS	<u>d2</u>	linear-equations
DSINDG	<u>c</u>	elementary-functions, special-functions
DSISL	<u>linpack</u>	symmetric
DSJAC	<u>d2</u>	linear-equations
DSLI	<u>d2</u>	linear-equations
DSLI2	<u>d2</u>	linear-equations
DSLITI	<u>d2</u>	linear-equations
DSLUBC	<u>d2</u>	linear-equations
DSLUCN	<u>d2</u>	linear-equations
DSLUCS	<u>d2</u>	linear-equations
DSLUGM	<u>d2</u>	linear-equations
DSLUI	<u>d2</u>	linear-equations
DSLUI2	<u>d2</u>	linear-equations
DSLUI4	<u>d2</u>	linear-equations
DSLUOM	<u>d2</u>	linear-equations
DSLUTI	<u>d2</u>	linear-equations
DSMMI2	<u>d2</u>	linear-equations
DSMMTI	<u>d2</u>	linear-equations
DSMTV	<u>d1b</u>	matrix-operations
DSMV	<u>d1b</u>	matrix-operations
DSORT	<u>n</u>	data-handling
DSOS	<u>f</u>	nonlinear-equations
DSPCO	<u>linpack</u>	symmetric
DSPDI	<u>linpack</u>	symmetric
DSPENC	<u>c</u>	elementary-functions, special-functions
DSPFA	<u>linpack</u>	symmetric
DSPLP	<u>g</u>	optimization
DSPMV	<u>d1b</u>	matrix-operations
DSPR	<u>d1b</u>	matrix-operations
DSPR2	<u>d1b</u>	matrix-operations
DSPSL	<u>linpack</u>	symmetric
DSTEPS	<u>i1</u>	ordinary-differential-equations
DSVDC	<u>d6</u>	singular-value-decomposition
DSWAP	<u>d1a</u>	vector-operations
DSYMM	<u>d1b</u>	matrix-operations
DSYMV	<u>d1b</u>	matrix-operations
DSYR	<u>d1b</u>	matrix-operations
DSYR2	<u>d1b</u>	matrix-operations
DSYR2K	<u>d1b</u>	matrix-operations
DSYRK	<u>d1b</u>	matrix-operations
DTBMV	<u>d1b</u>	matrix-operations
DTBSV	<u>d1b</u>	matrix-operations
DTIN	<u>n</u>	data-handling
DTOUT	<u>n</u>	data-handling
DTPMV	<u>d1b</u>	matrix-operations
DTPSV	<u>d1b</u>	matrix-operations
DTRCO	<u>linpack</u>	triangular
DTRDI	<u>linpack</u>	triangular
DTRMM	<u>d1b</u>	matrix-operations

DTRMV	dlb	matrix-operations
DTRSL	linpack	triangular
DTRSM	dlb	matrix-operations
DTRSV	dlb	matrix-operations
DULSIA	d9	overdetermined-systems, least-squares
DWNNLS	k	approximation
DXADD	a	arithmetic-functions
DXADJ	a	arithmetic-functions
DXC210	a	arithmetic-functions
DXCON	a	arithmetic-functions
DXLEGF	c	elementary-functions, special-functions
DXNRMP	c	elementary-functions, special-functions
DXRED	a	arithmetic-functions
DXSET	a	arithmetic-functions
E1	c	elementary-functions, special-functions
EFC	k	approximation
EI	c	elementary-functions, special-functions
EISDOC	d4	eigenvalues, eigenvectors
EISDOC	z	documentation
ELMBAK	eispack	
ELMHES	eispack	
ELTRAN	eispack	
ERF	c	elementary-functions, special-functions
ERFC	c	elementary-functions, special-functions
EXINT	c	elementary-functions, special-functions
EXPREL	c	elementary-functions, special-functions
EZFFTB	j1	fast-fourier-transforms
EZFFTF	j1	fast-fourier-transforms
EZFFTI	j1	fast-fourier-transforms
FAC	c	elementary-functions, special-functions
FC	k	approximation
FDUMP	r3	diagnostics, error-handling
FFTDQC	z	documentation
FIG1	eispack	
FIG12	eispack	
FUNDOC	c	elementary-functions, special-functions
FZERO	f	nonlinear-equations
GAMI	c	elementary-functions, special-functions
GAMIC	c	elementary-functions, special-functions
GAMIT	c	elementary-functions, special-functions
GAMLIM	c	elementary-functions, special-functions
GAMMA	c	elementary-functions, special-functions
GAMR	c	elementary-functions, special-functions
GAUS8	h2	quadrature, definite-integrals
GENBUN	i2	partial-differential-equations
HFTI	d9	overdetermined-systems, least-squares
HQR	eispack	
HQR2	eispack	
HSTCRT	i2	partial-differential-equations
HSTCSP	i2	partial-differential-equations
HSTCYL	i2	partial-differential-equations
HSTPLR	i2	partial-differential-equations
HSTSSP	i2	partial-differential-equations
HTRIB3	eispack	
HTRIBK	eispack	
HTRID3	eispack	
HTRIDI	eispack	
HW3CRT	i2	partial-differential-equations
HWSCRT	i2	partial-differential-equations
HWSCSP	i2	partial-differential-equations

HWSCYL	<u>i2</u>	partial-differential-equations
HWSPLR	<u>i2</u>	partial-differential-equations
HWSSSP	<u>i2</u>	partial-differential-equations
ICAMAX	<u>d1a</u>	vector-operations
ICOPY	<u>d1a</u>	vector-operations
IDAMAX	<u>d1a</u>	vector-operations
IMTQL1	<u>eispack</u>	
IMTQL2	<u>eispack</u>	
IMTQLV	<u>eispack</u>	
INITDS	<u>c</u>	elementary-functions, special-functions
INITS	<u>c</u>	elementary-functions, special-functions
INTRV	<u>e</u>	interpolation
INVIT	<u>eispack</u>	
IPPERM	<u>n</u>	data-handling
IPSORT	<u>n</u>	data-handling
ISAMAX	<u>d1a</u>	vector-operations
LLSIA	<u>d9</u>	overdetermined-systems, least-squares
LSEI	<u>k</u>	approximation
MINFIT	<u>eispack</u>	
NUMXER	<u>r3</u>	diagnostics, error-handling
ORTBAK	<u>eispack</u>	
ORTHES	<u>eispack</u>	
ORTRAN	<u>eispack</u>	
PCHBS	<u>e</u>	interpolation
PCHCM	<u>e</u>	interpolation
PCHDOC	<u>e</u>	interpolation
PCHDOC	<u>z</u>	documentation
PCHFD	<u>e</u>	interpolation
PCHFE	<u>e</u>	interpolation
PCHIA	<u>e</u>	interpolation
PCHIC	<u>e</u>	interpolation
PCHID	<u>e</u>	interpolation
PCHIM	<u>e</u>	interpolation
PCHSP	<u>e</u>	interpolation
PCOEF	<u>k</u>	approximation
PFQAD	<u>e</u>	interpolation
POCH	<u>c</u>	elementary-functions, special-functions
POCH1	<u>c</u>	elementary-functions, special-functions
POIS3D	<u>i2</u>	partial-differential-equations
POISTG	<u>i2</u>	partial-differential-equations
POLCOF	<u>e</u>	interpolation
POLFIT	<u>k</u>	approximation
POLINT	<u>e</u>	interpolation
POLYVL	<u>e</u>	interpolation
PPQAD	<u>e</u>	interpolation
PPVAL	<u>e</u>	interpolation
PSI	<u>c</u>	elementary-functions, special-functions
PSIFN	<u>c</u>	elementary-functions, special-functions
PVALUE	<u>k</u>	approximation
QAG	<u>h2</u>	quadrature, definite-integrals
QAGE	<u>h2</u>	quadrature, definite-integrals
QAGI	<u>h2</u>	quadrature, definite-integrals
QAGIE	<u>h2</u>	quadrature, definite-integrals
QAGP	<u>h2</u>	quadrature, definite-integrals
QAGPE	<u>h2</u>	quadrature, definite-integrals
QAGS	<u>h2</u>	quadrature, definite-integrals
QAGSE	<u>h2</u>	quadrature, definite-integrals
QAWC	<u>h2</u>	quadrature, definite-integrals
QAWCE	<u>h2</u>	quadrature, definite-integrals
QAWF	<u>h2</u>	quadrature, definite-integrals

QAWFE	<u>h2</u>	quadrature, definite-integrals
QAWO	<u>h2</u>	quadrature, definite-integrals
QAWOE	<u>h2</u>	quadrature, definite-integrals
QAWS	<u>h2</u>	quadrature, definite-integrals
QAWSE	<u>h2</u>	quadrature, definite-integrals
QC25C	<u>h2</u>	quadrature, definite-integrals
QC25F	<u>h2</u>	quadrature, definite-integrals
QC25S	<u>h2</u>	quadrature, definite-integrals
QK15	<u>h2</u>	quadrature, definite-integrals
QK15I	<u>h2</u>	quadrature, definite-integrals
QK15W	<u>h2</u>	quadrature, definite-integrals
QK21	<u>h2</u>	quadrature, definite-integrals
QK31	<u>h2</u>	quadrature, definite-integrals
QK41	<u>h2</u>	quadrature, definite-integrals
QK51	<u>h2</u>	quadrature, definite-integrals
QK61	<u>h2</u>	quadrature, definite-integrals
QMOMO	<u>h2</u>	quadrature, definite-integrals
QNC79	<u>h2</u>	quadrature, definite-integrals
QNG	<u>h2</u>	quadrature, definite-integrals
QPDOC	<u>h2</u>	quadrature, definite-integrals
QPDOC	<u>z</u>	documentation
QZHES	<u>eispack</u>	
QZIT	<u>eispack</u>	
QZVAL	<u>eispack</u>	
QZVEC	<u>eispack</u>	
R1MACH	<u>r1</u>	machine-constants
R9PAK	<u>a</u>	arithmetic-functions
R9UPAK	<u>a</u>	arithmetic-functions
RAND	<u>l</u>	pseudo-random-numbers
RATQR	<u>eispack</u>	
RC	<u>c</u>	elementary-functions, special-functions
RC3JJ	<u>c</u>	elementary-functions, special-functions
RC3JM	<u>c</u>	elementary-functions, special-functions
RC6J	<u>c</u>	elementary-functions, special-functions
RD	<u>c</u>	elementary-functions, special-functions
REBAK	<u>eispack</u>	
REBAKB	<u>eispack</u>	
REDUC	<u>eispack</u>	
REDUC2	<u>eispack</u>	
RF	<u>c</u>	elementary-functions, special-functions
RFFFTB1	<u>j1</u>	fast-fourier-transforms
RFFTF1	<u>j1</u>	fast-fourier-transforms
RFFTI1	<u>j1</u>	fast-fourier-transforms
RG	<u>eispack</u>	
RGAUSS	<u>l</u>	pseudo-random-numbers
RGG	<u>eispack</u>	
RJ	<u>c</u>	elementary-functions, special-functions
RPQR79	<u>f</u>	nonlinear-equations
RPZERO	<u>f</u>	nonlinear-equations
RS	<u>eispack</u>	
RSB	<u>eispack</u>	
RSG	<u>eispack</u>	
RSGAB	<u>eispack</u>	
RSGBA	<u>eispack</u>	
RSP	<u>eispack</u>	
RST	<u>eispack</u>	
RT	<u>eispack</u>	
RUNIF	<u>l</u>	pseudo-random-numbers
SASUM	<u>d1a</u>	vector-operations
SAXPY	<u>d1a</u>	vector-operations

SBCG	<u>d2</u>	linear-equations
SBHIN	<u>n</u>	data-handling
SBHIN	<u>n</u>	data-handling
SBOCLS	<u>k</u>	approximation
SBOLS	<u>k</u>	approximation
SCASUM	<u>d1a</u>	vector-operations
SCG	<u>d2</u>	linear-equations
SCGN	<u>d2</u>	linear-equations
SCGS	<u>d2</u>	linear-equations
SCHDC	<u>linpack</u>	cholesky-operations
SCHDD	<u>linpack</u>	cholesky-operations
SCHEX	<u>linpack</u>	cholesky-operations
SCHUD	<u>linpack</u>	cholesky-operations
SCNRM2	<u>d1a</u>	vector-operations
SCOPY	<u>d1a</u>	vector-operations
SCOPYM	<u>d1a</u>	vector-operations
SCOV	<u>k</u>	approximation
SCPPLT	<u>n</u>	data-handling
SCPPLT	<u>n</u>	data-handling
SDASSL	<u>i1</u>	ordinary-differential-equations
SDOT	<u>d1a</u>	vector-operations
SDRIV1	<u>i1</u>	ordinary-differential-equations
SDRIV2	<u>i1</u>	ordinary-differential-equations
SDRIV3	<u>i1</u>	ordinary-differential-equations
SDSDOT	<u>d1a</u>	vector-operations
SEPCLI	<u>i2</u>	partial-differential-equations
SEPX4	<u>i2</u>	partial-differential-equations
SGBCO	<u>linpack</u>	general-band
SGBDI	<u>linpack</u>	general-band
SGBFA	<u>linpack</u>	general-band
SGBSL	<u>linpack</u>	general-band
SGECO	<u>linpack</u>	general
SGEDI	<u>linpack</u>	general
SGEEV	<u>d4</u>	eigenvalues, eigenvectors
SGEFA	<u>linpack</u>	general
SGEFS	<u>d2</u>	linear-equations
SGEIR	<u>d2</u>	linear-equations
SGEMM	<u>d1b</u>	matrix-operations
SGEMV	<u>d1b</u>	matrix-operations
SGER	<u>d1b</u>	matrix-operations
SGESL	<u>linpack</u>	general
SGLSS	<u>d9</u>	overdetermined-systems, least-squares
SGMRES	<u>d2</u>	linear-equations
SGTSL	<u>linpack</u>	general-tridiagonal
SINDG	<u>c</u>	elementary-functions, special-functions
SINQB	<u>j1</u>	fast-fourier-transforms
SINQF	<u>j1</u>	fast-fourier-transforms
SINQI	<u>j1</u>	fast-fourier-transforms
SINT	<u>j1</u>	fast-fourier-transforms
SINTI	<u>j1</u>	fast-fourier-transforms
SINTRP	<u>i1</u>	ordinary-differential-equations
SIR	<u>d2</u>	linear-equations
SLLT12	<u>d2</u>	linear-equations
SLPDOC	<u>d2</u>	linear-equations
SNBCO	<u>d2</u>	linear-equations
SNBDI	<u>d3</u>	determinants
SNBFA	<u>d2</u>	linear-equations
SNBFS	<u>d2</u>	linear-equations
SNBIR	<u>d2</u>	linear-equations
SNBSL	<u>d2</u>	linear-equations

SNLS1	<u>k</u>	approximation
SNLS1E	<u>k</u>	approximation
SNRM2	<u>d1a</u>	vector-operations
SNSQ	<u>f</u>	nonlinear-equations
SNSQE	<u>f</u>	nonlinear-equations
SOMN	<u>d2</u>	linear-equations
SOS	<u>f</u>	nonlinear-equations
SPBCO	<u>linpack</u>	hermitian-positive-definite-band
SPBDI	<u>linpack</u>	hermitian-positive-definite-band
SPBFA	<u>linpack</u>	hermitian-positive-definite-band
SPBSL	<u>linpack</u>	hermitian-positive-definite-band
SPENC	<u>c</u>	elementary-functions, special-functions
SPLP	<u>g</u>	optimization
SPOCO	<u>linpack</u>	hermitian-positive-definite
SPODI	<u>linpack</u>	hermitian-positive-definite
SPOFA	<u>linpack</u>	hermitian-positive-definite
SPOFS	<u>d2</u>	linear-equations
SPOIR	<u>d2</u>	linear-equations
SPOSL	<u>linpack</u>	hermitian-positive-definite
SPPCO	<u>linpack</u>	hermitian-positive-definite
SPPDI	<u>linpack</u>	hermitian-positive-definite
SPPERM	<u>n</u>	data-handling
SPPFA	<u>linpack</u>	hermitian-positive-definite
SPPSL	<u>linpack</u>	hermitian-positive-definite
SPSORT	<u>n</u>	data-handling
SPTSL	<u>linpack</u>	positive-definite-tridiagonal
SQRDC	<u>d5</u>	qr-decomposition
SQRS�	<u>d5</u>	qr-decomposition
SROT	<u>d1a</u>	vector-operations
SROTG	<u>d1a</u>	vector-operations
SROTM	<u>d1a</u>	vector-operations
SROTMG	<u>d1a</u>	vector-operations
SS2LT	<u>d2</u>	linear-equations
SS2Y	<u>d1b</u>	matrix-operations
SSBMV	<u>d1b</u>	matrix-operations
SSCAL	<u>d1a</u>	vector-operations
SSD2S	<u>d2</u>	linear-equations
SSDBCГ	<u>d2</u>	linear-equations
SSDCG	<u>d2</u>	linear-equations
SSDCGN	<u>d2</u>	linear-equations
SSDCGS	<u>d2</u>	linear-equations
SSDGMR	<u>d2</u>	linear-equations
SSDI	<u>d1b</u>	matrix-operations
SSDOMN	<u>d2</u>	linear-equations
SSDS	<u>d2</u>	linear-equations
SSDSCL	<u>d2</u>	linear-equations
SSGS	<u>d2</u>	linear-equations
SSICGC	<u>d2</u>	linear-equations
SSICO	<u>linpack</u>	symmetric
SSICS	<u>d2</u>	linear-equations
SSIDI	<u>linpack</u>	symmetric
SSIEV	<u>d4</u>	eigenvalues, eigenvectors
SSIFA	<u>linpack</u>	symmetric
SSILUR	<u>d2</u>	linear-equations
SSILUS	<u>d2</u>	linear-equations
SSISL	<u>linpack</u>	symmetric
SSJAC	<u>d2</u>	linear-equations
SSLI	<u>d2</u>	linear-equations
SSLI2	<u>d2</u>	linear-equations
SSLITI	<u>d2</u>	linear-equations

SSLUBC	<u>d2</u>	linear-equations
SSLUCN	<u>d2</u>	linear-equations
SSLUCS	<u>d2</u>	linear-equations
SSLUGM	<u>d2</u>	linear-equations
SSLUI	<u>d2</u>	linear-equations
SSLUI2	<u>d2</u>	linear-equations
SSLUI4	<u>d2</u>	linear-equations
SSLUOM	<u>d2</u>	linear-equations
SSLUTI	<u>d2</u>	linear-equations
SSMMI2	<u>d2</u>	linear-equations
SSMMTI	<u>d2</u>	linear-equations
SSMTV	<u>d1b</u>	matrix-operations
SSMV	<u>d1b</u>	matrix-operations
SSORT	<u>n</u>	data-handling
SSPCO	<u>linpack</u>	symmetric
SSPDI	<u>linpack</u>	symmetric
SSPEV	<u>d4</u>	eigenvalues, eigenvectors
SSPFA	<u>linpack</u>	symmetric
SSPMV	<u>d1b</u>	matrix-operations
SSPR	<u>d1b</u>	matrix-operations
SSPR2	<u>d1b</u>	matrix-operations
SSPSL	<u>linpack</u>	symmetric
SSVDC	<u>d6</u>	singular-value-decomposition
SSWAP	<u>d1a</u>	vector-operations
SSYMM	<u>d1b</u>	matrix-operations
SSYMV	<u>d1b</u>	matrix-operations
SSYR	<u>d1b</u>	matrix-operations
SSYR2	<u>d1b</u>	matrix-operations
SSYR2K	<u>d1b</u>	matrix-operations
SSYRK	<u>d1b</u>	matrix-operations
STBMV	<u>d1b</u>	matrix-operations
STBSV	<u>d1b</u>	matrix-operations
STEPS	<u>i1</u>	ordinary-differential-equations
STIN	<u>n</u>	data-handling
STOUT	<u>n</u>	data-handling
STPMV	<u>d1b</u>	matrix-operations
STPSV	<u>d1b</u>	matrix-operations
STRCO	<u>linpack</u>	triangular
STRDI	<u>linpack</u>	triangular
STRMM	<u>d1b</u>	matrix-operations
STRMV	<u>d1b</u>	matrix-operations
STRSL	<u>linpack</u>	triangular
STRSM	<u>d1b</u>	matrix-operations
STRSV	<u>d1b</u>	matrix-operations
TINVIT	<u>eispack</u>	
TQL1	<u>eispack</u>	
TQL2	<u>eispack</u>	
TQLRAT	<u>eispack</u>	
TRBAK1	<u>eispack</u>	
TRBAK3	<u>eispack</u>	
TRED1	<u>eispack</u>	
TRED2	<u>eispack</u>	
TRED3	<u>eispack</u>	
TRIDIB	<u>eispack</u>	
TSTURM	<u>eispack</u>	
ULSIA	<u>d9</u>	overdetermined-systems, least-squares
WNNLS	<u>k</u>	approximation
XADD	<u>a</u>	arithmetic-functions
XADJ	<u>a</u>	arithmetic-functions
XC210	<u>a</u>	arithmetic-functions

XCON	<u>a</u>	arithmetic-functions
XERCLR	<u>r3</u>	diagnostics, error-handling
XERDMP	<u>r3</u>	diagnostics, error-handling
XERMAX	<u>r3</u>	diagnostics, error-handling
XGETF	<u>r3</u>	diagnostics, error-handling
XGETUA	<u>r3</u>	diagnostics, error-handling
XGETUN	<u>r3</u>	diagnostics, error-handling
XLEGF	<u>c</u>	elementary-functions, special functions
XNRMP	<u>c</u>	elementary-functions, special functions
XRED	<u>a</u>	arithmetic-functions
XSET	<u>a</u>	arithmetic-functions
XSETF	<u>r3</u>	diagnostics, error-handling
XSETUA	<u>r3</u>	diagnostics, error-handling
XSETUN	<u>r3</u>	diagnostics, error-handling
ZAIRY	<u>c</u>	elementary-functions, special-functions
ZBESH	<u>c</u>	elementary-functions, special-functions
ZBESI	<u>c</u>	elementary-functions, special-functions
ZBESJ	<u>c</u>	elementary-functions, special-functions
ZBESK	<u>c</u>	elementary-functions, special-functions
ZBESY	<u>c</u>	elementary-functions, special-functions
ZBIRY	<u>c</u>	elementary-functions, special-functions

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government thereof, and shall not be used for advertising or product endorsement purposes. (C) Copyright 1996 The Regents of the University of California. All rights reserved.

Keyword Index

The major sections of the SLATEC1 document have these keywords:

Keyword	Description
<u>entire</u>	The whole SLATEC1 document.
<u>title</u>	The name of this document.
<u>who</u>	Who to contact for help with SLATEC.
<u>introduction</u>	Background on the SLATEC library.
<u>slatec-documentation</u>	Structure of all local SLATEC docs.
<u>error-procedure</u>	How SLATEC subroutines report errors.
<u>b-spline-background</u>	Handling of B-splines by SLATEC routines.
<u>eispack-background</u>	Technical details on EISPACK routines.
<u>special-functions-background</u>	Comparative details on spec-function routines.
<u>quadpack-background</u>	Technical details on QUADPACK integrators.
<u>quadpack-routines</u>	Survey of included routines.
<u>quadpack-guidelines</u>	Suggestions for using QUADPACK well.
<u>quadpack-examples</u>	Sample calls to QUADPACK routines.
<u>pchip-background</u>	Tech. details on piecewise cubic Hermite interpolation package.
<u>pchip-routines</u>	Survey of included routines.
<u>pchip-references</u>	Background on PCHIP algorithms.
<u>prologue-format</u>	How to read SLATEC internal prologues.
<u>categories</u>	List of subject categories and codes used for SLATEC subroutines (links outward).
<u>routines</u>	Brief descriptions of SLATEC subroutines grouped by functional category.
<u>subroutine-dictionary</u>	Alphabetical list of SLATEC subroutines showing subject category and function.
<u>index</u>	This index of topics covered.
<u>revisions</u>	SLATEC1 revision history.
<u>date</u>	Latest revisions to SLATEC1.

To help you find relevant subroutines, SLATEC1 contains these special retrieval aids:

Keyword	Description
<u>categories</u>	List of subject categories and codes used for SLATEC subroutines (links outward).
<u>routines</u>	Brief descriptions of SLATEC subroutines grouped by functional category.
<u>subroutine-dictionary</u>	Alphabetical list of SLATEC subroutines showing subject category and function.

To see brief descriptions of the SLATEC routines in any subject category, use any of the GAMS category codes listed here as links to the corresponding section:

Category Keyword -----	Functional Category Description -----
<u>a</u>	arithmetic-functions
<u>c</u>	elementary-functions, special-functions
<u>d1a</u>	vector-operations
<u>d1b</u>	matrix-operations
<u>d2</u>	linear-equations
<u>d3</u>	determinants
<u>d4</u>	eigenvalues, eigenvectors
<u>d5</u>	qr-decomposition
<u>d6</u>	singular-value-decomposition
<u>d9</u>	overdetermined-systems, least-squares
<u>e</u>	interpolation
<u>f</u>	nonlinear-equations
<u>g</u>	optimization
<u>h2</u>	quadrature, definite-integrals
<u>i1</u>	ordinary-differential-equations
<u>i2</u>	partial-differential-equations
<u>j1</u>	fast-fourier-transforms
<u>k</u>	approximation
<u>l</u>	statistics, pseudo-random-numbers
<u>n</u>	data-handling
<u>r1</u>	machine-constants
<u>r3</u>	diagnostics, error-handling
<u>z</u>	documentation
<u>eispack</u>	Eigenvalue, eigenvector problems, like D4
<u>linpack</u>	Linear-algebra matrix solvers for these types: <u>cholesky-operations</u> <u>complex-hermitian</u> <u>general</u> <u>general-band</u> <u>general-tridiagonal</u> <u>hermitian-positive-definite</u> <u>hermitian-positive-definite-band</u> <u>positive-definite-tridiagonal</u> <u>symmetric</u> <u>triangular</u>

Date and Revisions

Revision date	Keyword affected	Description of changes
-----	-----	-----
26Feb96	<u>entire</u> <u>prologue-format</u> <u>slatec-documentation</u> <u>routines</u>	SLATEC1 overhauled for version 4.1. New section added. Manuals reorganized, subdivided. 200 more routines added.
07Nov91	a c e r3 index introduction subroutine-dictionary who	Added arithmetic functions XADD, DXADD, XADJ, DXADJ, XCON, and DXCON. Added special functions XLEGF, DXLEGF, XNRMP, DXNRMP. Added interpolation routines PCHCM, DPCHCM. Added error-handling routine XERMSG. Revised split between SLATEC2, SLATEC3. Revised split between SLATEC2, SLATEC3. New routines added alphabetically. Obsolete routines deleted from list. New SLATEC support contact.
30Nov87	entire	SLATEC1 overhauled for version 3.1. New category list, new routine summaries, new multitasking section.
22Aug83	documentation	Writeup scope and status clarified.
20Oct82	entire	First edition of new writeup.

TRG (26Feb96)

UCID-19631,19632,19633

Privacy and Legal Notice (URL: <http://www.llnl.gov/disclaimer.html>)

TRG (26Feb96) Contact on the OCF: lc-hotline@llnl.gov, on the SCF: lc-hotline@pop.llnl.gov