NCAR Software Catalog

1986 Edition

Russell Rew
Dick Valent
Harsh Anand Passi
Ginger Caldwell
PREFACE

The NCAR Software Catalog has existed as a set of on-line files available to users of NCAR computers since September 1980. This second edition of the published version incorporates several improvements over the first edition:

- a refinement of the categorization scheme (from 120 to 211 software categories), to make it easier to locate software;
- use of a stream editor and an equation preprocessor to incorporate mathematical notation into the software descriptions in a way that can be maintained with future changes to the descriptions;
- use of a laser printer to improve the readability of the resulting document;
- a significant increase in the number of library routines included in the catalog (from 2150 to 3343 user-level routines);
- inclusion of some operating system commands, statistical package subcommands, and utility software programs, since these can be used to accomplish some tasks more easily than writing a program; and
- elimination of the volatile entry-name index, since this is primarily used by library consultants and maintainers rather than users.

Most of the libraries have expanded modestly since the last edition, several old libraries have been merged, and several new libraries have been acquired.

This published version of the software catalog cannot be revised as frequently as the on-line text files, which are updated whenever a library package is acquired or deleted. Therefore this document should be considered a snapshot of the state of NCAR’s public software libraries as of February 1986. Users who need the most up-to-date information about available software should consult the on-line version of this catalog.

This software catalog does not include comparisons of software packages within categories or overviews of problem areas. Fortunately for the user, there are fine guides to various problem areas in the chapter introductions of the NAG [1], PORT [2], and IMSL [3] library manuals; other excellent sources of information about available mathematical software include [4], [5], and [6].

The present catalog owes much to other sources. In particular, the current classification scheme was originally based on [7], and has been revised to agree more closely with the classification scheme used in [4]. Eventually, we would like to move closer to a “facet” scheme such as is described in [8]. The brief routine descriptions in the category summaries have been taken or adapted from documentation provided with the libraries.

NCAR
Boulder, Colorado
February 1986

Russell Rew
Dick Valent
Harsh Anand Passi
Ginger Caldwell


OVERVIEW

INTRODUCTION

One way to enhance the usefulness of computers for scientific research is to organize commonly needed computer programs into software libraries. A software library is a collection of routines that may be maintained and accessed as a unit, and that share common conventions for documentation and use. An extensive collection of library software is available for use on the Scientific Computing Division (SCD) computers at NCAR, and is cataloged in this document.

The NCAR Software Catalog is intended to assist users of the SCD computers in locating software appropriate to their problems, and to collect in one place all the information about library access, documentation, and support for each of the software libraries available at NCAR.

This Catalog is not a substitute for vendor programming language manuals, operating system documentation, or vendor library reference manuals. The Libraries section of the Catalog (the last section) describes where to find detailed documentation or how to access online information for particular libraries.

Why So Many Libraries?

There are currently about 20 supported libraries for SCD's mainframe computers containing over 8000 routines. Of these, over 3300 routines are user-level, and descriptions of each of these are included in the NCAR Software Catalog under one or more of 207 software categories. In addition, subcommands of the SAS statistical package (which is not a library) and utility programs (which are not library subroutines) are included in this Catalog as if they were library subroutines, to help users locate such software.

Most of this software has been acquired from sources outside NCAR, while some has been locally developed. When first confronted with NCAR's large collection of public library software, users may wonder why all the existing libraries have not been merged into a single library that contains the best available routine for each problem.

There are several reasons why more than one library is supported and the source of routines from different collections is not made transparent to users. To select only the best implementations of the best algorithms for a single library would require almost continuous reevaluation of routines in each area. Often there is no routine or algorithm that is clearly best for solving a particular type of problem; different routines implement different compromises between accuracy, efficiency, memory requirements, robustness, ease of use, flexibility, and generality. Furthermore, since the problem domain of most routines is infinite, it is impossible to test competing routines that use different methods and determine which is best for each problem, even for a narrowly defined performance measure.
collisions and maintenance considerations in a multimachine environment also make it impractical to merge all available libraries into a single super-library.

The name of the library to which a routine belongs is important information that identifies the source of the routine, the conventions for documentation and use of the routine, and the level of support provided to users. Two problems associated with keeping libraries separate are the inconvenience to users who must know how different libraries are accessed to use them, and the difficulty in locating a routine for a particular problem from many library manuals that all use different indexing schemes. The NCAR Software Catalog is a partial remedy to these problems: it collects all the information from various separate sources that users need to know to access routines or documentation from any library, and it classifies all available library routines by a uniform set of categories.

SCD Software Libraries

Library packages available for use include many common numerical algorithms, graphical display utilities, input/output packages, and other kinds of software tools. Libraries supported at various levels on the SCD's Cray computers include three commercial libraries (PORT from Bell Laboratories, NAG from the Numerical Algorithms Group, and IMSL from International Mathematical and Statistical Libraries, Inc.), libraries supported by Cray Research ($SCILIB and $SYSLIB), some acquired collections of special-purpose mathematical software (AMOSLIB, EDA, EISPACK, EISPKD, FUNPACK, LINPACK, ITPACK, FITPACK, MINPACK, SLATEC, STARPAC, STATLIB), and libraries containing locally developed software and software from other sources (LOCLIB, $NCARLB). In addition, the SAS statistical package is available to IBM 4341 users, so SAS commands have been included in this Catalog as if they were routines in a SAS “library”. Utilities and software tools from various sources have been included under the UTILITIES “library”.

These libraries are described in more detail in the last section of the Catalog, in individual library summaries. Each library summary tells the origin of the software, and describes where to get documentation for the library, how to access on-line documentation for individual library subroutines, how to access source or compiled versions of the library, how to access examples of use of each routine, and what level of support users can expect. A brief list of the purpose and source of each library follows:

AMOSLIB an extensive collection of special function software acquired from Sandia Laboratories.
EDA  a collection of statistical software for Exploratory Data Analysis.

EISPACK a collection of eigenvalue and eigenvector software acquired from the National Energy Software Center.

EISPKD a collection of double precision versions of eigenvalue and eigenvector software acquired from the National Energy Software Center.

FITPACK a collection of routines for fitting and approximation using splines under tension acquired from the University of Texas at Austin.

FUNPACK a collection of special function software acquired from the National Energy Software Center.

IMSL a general-purpose mathematical and statistical software library, leased from International Mathematical and Statistical Libraries for use on the Cray computers.

ITPACK a collection of subroutines for the iterative solution of linear systems acquired from the University of Texas at Austin.

LINPACK a collection of linear algebra software acquired from the National Energy Software Center.

LOCLIB a general-purpose library of locally maintained mathematical, graphics, and utility software, most noted for its locally developed graphics and partial differential equation packages. Parts of this library receive the highest level of support.

MINPACK a collection of subroutines for solving systems of nonlinear equations and for nonlinear least-squares problems acquired from the National Energy Software Center.

NAG a general-purpose mathematical and statistical software library leased from the Numerical Algorithms Group for the Cray computers.

$NCARLB a locally maintained Cray object library made from a subset of LOCLIB.

PORT a general-purpose mathematical and utility software library, developed by Bell Laboratories and purchased for use on the Cray computers from Western Electric.

SAS a large statistical analysis system leased from the SAS Institute for use on the IBM 4341 front end computers.
Overview

SCILIB a collection of scientific and mathematical subroutines for the Cray-1 and Cray XMP, developed and maintained by Cray Research, Inc.

SLATEC a large collection of portable public domain software supported at Department of Energy laboratories.

STARPAC a collection of statistical analysis subroutines from the Statistical Engineering Division of the National Bureau of Standards.

STATLIB a collection of statistical analysis subroutines from the Statistical Engineering Division of the National Bureau of Standards, which is gradually being superceded by STARPAC routines.

SYSLIB a library of system utilities for the Cray computers, developed and maintained by Cray Research, Inc.

UTILITIES a pseudo-collection of Cray utilities and software tools, which are not really elements of a library but are described as if they were for convenience in this Catalog.

HOW TO USE THIS CATALOG

The NCAR Software Catalog is intended to help you quickly locate software in the various libraries. All supported software libraries above the level of the compiler primitives are covered included. The Catalog contains a list of software categories and, within each category, short descriptions of all the library packages belonging to that category. A separate section summarizes the purpose, access, documentation, and support level for each library.

To find out what software is available for a particular problem, follow the steps outlined below.

1. Determine which software category or categories are appropriate. A keyword index of software categories follows this overview. Following that is another list of the same categories, ordered by category name. Although it is usually easier to find a category using the keyword list, the list by category name may be more helpful for users who have become familiar with the category scheme or who want to examine several closely related categories. All categories that belong to the same major subject area begin with the same single upper-case letter.

For example, if a routine to solve ordinary differential equation boundary value problems is desired, you might look under “Boundary” in the alphabetized keyword list and find the entry Boundary D2b. Boundary Value Problems for Ordinary Differential Equations

Thus “D2b” is the category key for this subject area.

Software packages that solve problems in more than one category are included in each applicable category. If a problem
does not fit neatly into any single category, it may be necessary to check several categories for relevant software.

2. Find the desired category summary, listing the names and descriptions of all routines from all libraries that are appropriate for that category. Category summaries are listed alphabetically in the large section of the Catalog that follows the list of categories. For example, there is a section labeled “D2b” that describes routines from various libraries that may be used to solve different kinds of ordinary differential equation boundary value problems.

3. Determine whether any library software exists that is suitable. Each entry in a category file lists the name and short description of purpose for the library routine or package. The library name appears centered and italicized above the routine description. For example, one entry under the “D2b” section is

**D02RAF** Solves the two-point boundary-value problem with general boundary conditions for a system of ordinary differential equations, using a deferred correction technique and Newton iteration.

The short descriptions may be adequate to choose an appropriate routine, but you may also need to consult numerical analysis texts, local specialists, or chapter introductions in library manuals available in the SCD Consulting Office to understand some of the terminology used in the short descriptions.

The order in which the entries under any particular category are listed is determined solely by the libraries to which the routines belong. The libraries appear in alphabetical order, thus there is no implied judgement that the routines that appear first are any better than the routines that appear last. Users should reach each of the relevant descriptions to select which routines are the best candidates for their problems.

Even if no appropriate software appears in the category summary, it is possible that software is available that has not been implemented or tested at NCAR yet. Please check with a member of the Software and Libraries Group (the Consulting Office can direct queries to the appropriate person) or the Advanced Methods Group to determine if a package is available from an unsupported collection of software, or to request that NCAR acquire and support some software for a particular purpose.

4. If you have identified one or more candidate routines, you will next need to know how to get more information about how to use the routines. The key to this information is the library name associated with the routine. For example, “NAG” is the library name for the routine D02RAF in the above excerpt from the category summary for “D2b.”
Information common to all routines in a particular library including how to list on-line documentation, where to find manuals, how to access the routines on a particular computer, and what level of support can be expected for the routines may be found in the appropriate library summary. All the library summaries appear in alphabetical order following the category summaries. For example, the NAG library summary is included in the section with “NAG” as the page header, and includes all the information necessary to use NAG routines.

On-line Help

An on-line version of the NCAR Software Catalog is available as text files that contain the list of categories, category summaries, and library summaries. The on-line version is called ISORE (Inventory of Software REsources). As library collections grow and routines within each library are added, modified, and deleted, the on-line software catalog files are revised to reflect the current state of the collection. Categories will also be changed as the classification scheme is refined, so the on-line list of categories should be consulted when using the on-line catalog. Information on how to use the on-line version of the catalog may be found by issuing the command “HELP ISORE” from the IBM 4341 VM/CMS system.

About the Library Routine Descriptions

Astute readers may notice a lack of uniformity in the wording, punctuation, notation, and level of detail in library routine descriptions for different libraries. For example, IMSL routine descriptions are usually less detailed than NAG descriptions. These inconsistencies are unfortunate but necessary, given that the vendor-provided routine descriptions must be used as the basis for this Catalog rather than rewriting all the descriptions to make them uniform.

Changes to incorporate mathematical notation have been implemented using a set of stream editor scripts that produce modified descriptions from the original vendor-provided descriptions. Some spelling corrections and other simple changes have also been incorporated into these editor scripts. The implementation of this Catalog as a series of programs that operate on the vendor-provided descriptions augmented with category codes to produce input for a text-formatting program allows us to maintain only one version of the descriptions for both the on-line and published versions of the NCAR Software Catalog. This approach also ensures that the published version will accurately reflect the state of the libraries and makes it possible to provide timely updates to the on-line version when new routines are added to existing libraries.
The following keyword index to the categories used in the NCAR software classification scheme is designed to help quickly find the category code for a given problem. The first column contains keywords used in the category descriptions. The second column is the category code under which all relevant software is listed. On the right is the complete category description, to verify the relevance of the keyword in the context in which it appears.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA</td>
<td>L10c.</td>
<td>Time Series: ARMA and ARIMA Modeling and Forecasting</td>
</tr>
<tr>
<td>ARMA</td>
<td>L10c.</td>
<td>Time Series: ARMA and ARIMA Modeling and Forecasting</td>
</tr>
<tr>
<td>Access</td>
<td>J2.</td>
<td>Direct (Random) Access Input/Output</td>
</tr>
<tr>
<td>Access</td>
<td>K1.</td>
<td>File Management: Access, Deletion, and Staging</td>
</tr>
<tr>
<td>Aids</td>
<td>F1a8.</td>
<td>Other Elementary Vector Operations, Including Vectorization Aids</td>
</tr>
<tr>
<td>Airy</td>
<td>B51.</td>
<td>Miscellaneous Bessel and Related Functions, Airy Functions</td>
</tr>
<tr>
<td>Allocation</td>
<td>Q6.</td>
<td>Storage Allocation, Memory Management, and Overlays</td>
</tr>
<tr>
<td>Approximating</td>
<td>E8a.</td>
<td>Evaluation of Interpolating or Approximating Functions using B-Splines</td>
</tr>
<tr>
<td>Approximating</td>
<td>E8b.</td>
<td>Evaluation of Interpolating or Approximating Functions using Chebyshev Series</td>
</tr>
<tr>
<td>Approximating</td>
<td>E8c.</td>
<td>Evaluation of Interpolating or Approximating Functions using Cubic Splines</td>
</tr>
<tr>
<td>Approximating</td>
<td>E8d.</td>
<td>Evaluation of Interpolating or Approximating Functions using Splines under Tension</td>
</tr>
<tr>
<td>Approximation</td>
<td>E2a.</td>
<td>Linear Least Squares Approximation, Overdetermined Linear Systems (see also L08a3)</td>
</tr>
<tr>
<td>Approximation</td>
<td>E2b.</td>
<td>Nonlinear Least Squares Approximation (see also L08b)</td>
</tr>
<tr>
<td>Approximations</td>
<td>E5.</td>
<td>Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems</td>
</tr>
<tr>
<td>Category</td>
<td>Keyword Index</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Archiving</td>
<td>K7. File Archiving and Version Control</td>
<td></td>
</tr>
<tr>
<td>Arithmetic</td>
<td>A. Arithmetic, Elementary Operations on Polynomials</td>
<td></td>
</tr>
<tr>
<td>Asynchronous</td>
<td>J3. Asynchronous (Buffered) Input/Output</td>
<td></td>
</tr>
<tr>
<td>Autocorrelation</td>
<td>L10b. Time Series: Autocorrelation and Cross-Correlation, Univariate and Multivariate</td>
<td></td>
</tr>
<tr>
<td>Automatic</td>
<td>H2a3a1. One-Dimensional Integration: Finite Interval, Automatic, General Integrand</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H2a3a2. One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H2a3b1. One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H2a3b2. One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities</td>
<td></td>
</tr>
<tr>
<td>B-Splines</td>
<td>E8a. Evaluation of Interpolating or Approximating Functions using B-Splines</td>
<td></td>
</tr>
<tr>
<td>Banded</td>
<td>F2c8. Eigenvalues and Eigenvectors of Banded Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4a2. Linear Systems: Real Non-Symmetric Banded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4b2. Linear Systems: Real Symmetric Banded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4c2. Linear Systems: Complex Non-Hermitian Banded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4d2. Linear Systems: Complex Hermitian Banded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4e2. Linear Systems: Double-Precision Non-Symmetric Banded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4f2. Linear Systems: Double-Precision Symmetric Banded</td>
<td></td>
</tr>
<tr>
<td>Bessel</td>
<td>B5i1. I Bessel Functions of Real Argument</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B5i2. K Bessel Functions of Real Argument</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B5j1. J Bessel Functions of Real Argument</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B5j2. Y Bessel Functions of Real Argument</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B5k. Bessel Functions of Complex Argument, Kelvin Functions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B5l. Miscellaneous Bessel and Related Functions, Airy Functions</td>
<td></td>
</tr>
<tr>
<td>Beta</td>
<td>B5f. Gamma, Beta, Psi Functions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05a. Statistical Distributions: Beta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L06a. Pseudo-Random Number Generators: Beta</td>
<td></td>
</tr>
<tr>
<td>Binomial</td>
<td>L05b. Statistical Distributions: Binomial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05r. Statistical Distributions: Negative Binomial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L06b. Pseudo-Random Number Generators: Binomial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L06r. Pseudo-Random Number Generators: Negative Binomial</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>M4. Character Manipulation, Bit Operations</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td>D2b. Boundary Value Problems for Ordinary Differential Equations</td>
<td></td>
</tr>
<tr>
<td>Keyword</td>
<td>Index</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Bounds</td>
<td>I2h1.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
<tr>
<td>Buffered</td>
<td>J3.</td>
<td>Asynchronous (Buffered) Input/Output</td>
</tr>
<tr>
<td>Canonical</td>
<td>L13c.</td>
<td>Canonical Correlation Analysis</td>
</tr>
<tr>
<td>Categorical</td>
<td>L09.</td>
<td>Categorical Data Analysis (Discrete Data Analysis) (see also L01b)</td>
</tr>
<tr>
<td>Cauchy</td>
<td>L06c.</td>
<td>Pseudo-Random Number Generators: Cauchy</td>
</tr>
<tr>
<td>Character</td>
<td>M1a3.</td>
<td>Sorting of Character Data</td>
</tr>
<tr>
<td></td>
<td>M4.</td>
<td>Character Manipulation, Bit Operations</td>
</tr>
<tr>
<td>Characters</td>
<td>J5a5.</td>
<td>Graphics: Writing Characters</td>
</tr>
<tr>
<td>Chebyshev</td>
<td>E8b.</td>
<td>Evaluation of Interpolating or Approximating Functions using Chebyshev Series</td>
</tr>
<tr>
<td>Chi-squared</td>
<td>L05d.</td>
<td>Statistical Distributions: Chi-squared</td>
</tr>
<tr>
<td></td>
<td>L06d.</td>
<td>Pseudo-Random Number Generators: Chi-squared</td>
</tr>
<tr>
<td>Circular</td>
<td>L05e.</td>
<td>Statistical Distributions: Circular Coverage Function</td>
</tr>
<tr>
<td>Cluster</td>
<td>L14.</td>
<td>Cluster Analysis</td>
</tr>
<tr>
<td>Combination</td>
<td>K4.</td>
<td>File Combination, Splitting, and Conversion Tools</td>
</tr>
<tr>
<td>Communications</td>
<td>J6.</td>
<td>Miscellaneous I/O, Including Communications, Special Devices</td>
</tr>
<tr>
<td>Comparison</td>
<td>K6.</td>
<td>File Comparison</td>
</tr>
<tr>
<td>Complex</td>
<td>B5k.</td>
<td>Bessel Functions of Complex Argument, Kelvin Functions</td>
</tr>
<tr>
<td></td>
<td>F2c5.</td>
<td>Eigenvalues and Eigenvectors of Complex Hermitian Matrices</td>
</tr>
<tr>
<td></td>
<td>F2c6.</td>
<td>Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices</td>
</tr>
<tr>
<td></td>
<td>F4c1.</td>
<td>Linear Systems: Complex Non-Hermitian Full</td>
</tr>
<tr>
<td></td>
<td>F4c2.</td>
<td>Linear Systems: Complex Non-Hermitian Banded</td>
</tr>
<tr>
<td></td>
<td>F4d1a.</td>
<td>Linear Systems: Complex Hermitian Full Indefinite</td>
</tr>
<tr>
<td></td>
<td>F4d1b.</td>
<td>Linear Systems: Complex Hermitian Full Positive-Definite</td>
</tr>
<tr>
<td></td>
<td>F4d2.</td>
<td>Linear Systems: Complex Hermitian Banded</td>
</tr>
<tr>
<td>Component</td>
<td>L13b.</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>Confidence</td>
<td>L04a.</td>
<td>Confidence Intervals, Hypothesis Testing</td>
</tr>
</tbody>
</table>

NCAR Software Catalog
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conformal</td>
<td>D7.</td>
<td>Conformal Mapping, Schwarz-Christoffel Transformations</td>
</tr>
<tr>
<td>Constants</td>
<td>Q3.</td>
<td>Machine and System Dependent Constants</td>
</tr>
<tr>
<td>Constraints</td>
<td>I2h1.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
<tr>
<td></td>
<td>I2h3.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints</td>
</tr>
<tr>
<td>Control</td>
<td>K7.</td>
<td>File Archiving and Version Control</td>
</tr>
<tr>
<td></td>
<td>K9.</td>
<td>File Characteristics and Permissions (Access Control)</td>
</tr>
<tr>
<td></td>
<td>Q7.</td>
<td>Job Control, Job Status, Operating System Requests</td>
</tr>
<tr>
<td>Conversion</td>
<td>F1b7.</td>
<td>Matrix Storage Mode Conversion</td>
</tr>
<tr>
<td></td>
<td>K4.</td>
<td>File Combination, Splitting, and Conversion Tools</td>
</tr>
<tr>
<td></td>
<td>M2.</td>
<td>Conversion, Packing and Unpacking</td>
</tr>
<tr>
<td>Convolutions</td>
<td>G2.</td>
<td>Convolutions</td>
</tr>
<tr>
<td>Copy</td>
<td>F1a3.</td>
<td>Vector Copy or Swap</td>
</tr>
<tr>
<td>Copying</td>
<td>K2.</td>
<td>File Copying, Renaming</td>
</tr>
<tr>
<td>Correlation</td>
<td>L11a.</td>
<td>Correlation Analysis: No Missing Values</td>
</tr>
<tr>
<td></td>
<td>L11b.</td>
<td>Correlation Analysis: Missing Values</td>
</tr>
<tr>
<td></td>
<td>L13c.</td>
<td>Canonical Correlation Analysis</td>
</tr>
<tr>
<td>Cosine</td>
<td>B5e.</td>
<td>Exponential, Logarithmic, Sine and Cosine Integrals</td>
</tr>
<tr>
<td></td>
<td>G1.</td>
<td>Fast Fourier Transforms, Sine and Cosine Transforms</td>
</tr>
<tr>
<td>Covariance</td>
<td>L07d.</td>
<td>Analysis of Covariance</td>
</tr>
<tr>
<td></td>
<td>L07f.</td>
<td>Tests to use with Analysis of Variance and Covariance</td>
</tr>
<tr>
<td>Coverage</td>
<td>L05e.</td>
<td>Statistical Distributions: Circular Coverage Function</td>
</tr>
<tr>
<td>Cross-Correlation</td>
<td>L10b.</td>
<td>Time Series: Autocorrelation and Cross-Correlation, Univariate and Multivariate</td>
</tr>
<tr>
<td>Cubic</td>
<td>E8c.</td>
<td>Evaluation of Interpolating or Approximating Functions using Cubic Splines</td>
</tr>
<tr>
<td>Dashed</td>
<td>J5a1.</td>
<td>Graphics: Plotting Dashed Lines</td>
</tr>
<tr>
<td>Date</td>
<td>Q1.</td>
<td>Date and Time, Program Timers</td>
</tr>
<tr>
<td>Debugging</td>
<td>N.</td>
<td>Debugging</td>
</tr>
<tr>
<td>Decomposition</td>
<td>F5.</td>
<td>QR (Orthogonal) Decomposition</td>
</tr>
<tr>
<td></td>
<td>F6.</td>
<td>Singular Value Decomposition, Rank Determination, Generalized Inverse</td>
</tr>
<tr>
<td>Category</td>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Decompositions</td>
<td>F7.</td>
<td>Updating Matrix Decompositions</td>
</tr>
<tr>
<td>Deletion</td>
<td>K1.</td>
<td>File Management: Access, Deletion, and Staging</td>
</tr>
<tr>
<td>Densities</td>
<td>L01b.</td>
<td>Descriptive Statistics: Distributions, Densities, Frequencies (see also L03a and L03e)</td>
</tr>
<tr>
<td>Dependent</td>
<td>Q3.</td>
<td>Machine and System Dependent Constants</td>
</tr>
<tr>
<td>Descriptive</td>
<td>L01a.</td>
<td>Descriptive Statistics: Location, Dispersion, Shape</td>
</tr>
<tr>
<td></td>
<td>L01b.</td>
<td>Descriptive Statistics: Distributions, Densities, Frequencies (see also L03a and L03e)</td>
</tr>
<tr>
<td>Designs</td>
<td>L07a.</td>
<td>Generation of Experimental Designs</td>
</tr>
<tr>
<td>Development</td>
<td>S.</td>
<td>Software Development Tools, Language Processors</td>
</tr>
<tr>
<td>Devices</td>
<td>J6.</td>
<td>Miscellaneous I/O, Including Communications, Special Devices</td>
</tr>
<tr>
<td>Differential</td>
<td>D2a1.</td>
<td>Initial Value Problems for Non-Stiff Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D2a2.</td>
<td>Initial Value Problems for Stiff Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D2b.</td>
<td>Boundary Value Problems for Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D2c.</td>
<td>Eigenvalue Problems for Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D3.</td>
<td>Partial Differential Equations</td>
</tr>
<tr>
<td>Differentiation</td>
<td>H1a.</td>
<td>Numerical Differentiation: Function Available only on Grid</td>
</tr>
<tr>
<td></td>
<td>H1b.</td>
<td>Numerical Differentiation: User-Defined Function</td>
</tr>
<tr>
<td>Direct</td>
<td>J2.</td>
<td>Direct (Random) Access Input/Output</td>
</tr>
<tr>
<td>Directory</td>
<td>K5.</td>
<td>File Directory, Status, and Size Information</td>
</tr>
<tr>
<td>Discontinuities</td>
<td>H2a3a2.</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td></td>
<td>H2a3b2.</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td>Discrete</td>
<td>L09.</td>
<td>Categorical Data Analysis (Discrete Data Analysis) (see also L01b)</td>
</tr>
<tr>
<td>Discriminant</td>
<td>L12.</td>
<td>Discriminant Analysis</td>
</tr>
<tr>
<td>Dispersion</td>
<td>L01a.</td>
<td>Descriptive Statistics: Location, Dispersion, Shape</td>
</tr>
<tr>
<td>Distribution-Free</td>
<td>L04b.</td>
<td>Nonparametric (Distribution-Free) Analysis (see also NCAR Software Catalog)</td>
</tr>
<tr>
<td>Category</td>
<td>Keywords</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Distributions</td>
<td>L07c, L11a, L11b) Descriptive Statistics: Distributions, Densities, Frequencies (see also L03a and L03e)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05a. Statistical Distributions: Beta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05b. Statistical Distributions: Binomial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05d. Statistical Distributions: Chi-squared</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05e. Statistical Distributions: Circular Coverage Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05i. Statistical Distributions: F Distribution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05j. Statistical Distributions: Gamma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05m. Statistical Distributions: Hypergeometric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05n. Statistical Distributions: Kolmogorov-Smirnov</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05r. Statistical Distributions: Negative Binomial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05s. Statistical Distributions: Normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05u. Statistical Distributions: Poisson</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05v. Statistical Distributions: T Distribution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L05y. Statistical Distributions: General Distribution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L06z. Pseudo-Random Number Generators: Other Distributions</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Q4. Documentation Retrieval</td>
<td></td>
</tr>
<tr>
<td>Dot</td>
<td>F1a2. Vector Dot Product</td>
<td></td>
</tr>
<tr>
<td>Double-Precision</td>
<td>F4e1. Linear Systems: Double-Precision Non-Symmetric Full</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4e2. Linear Systems: Double-Precision Non-Symmetric Banded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4f1a. Linear Systems: Double-Precision Symmetric Full Indefinite</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4f1b. Linear Systems: Double-Precision Symmetric Full Positive-Definite</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4f2. Linear Systems: Double-Precision Symmetric Banded</td>
<td></td>
</tr>
<tr>
<td>Editing</td>
<td>K3. File Editing</td>
<td></td>
</tr>
<tr>
<td>Eigenvalue</td>
<td>D2c. Eigenvalue Problems for Ordinary Differential Equations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2a1. Eigenvalue Problems: Reduction to Symmetric Tridiagonal Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2a2. Eigenvalue Problems: Reduction to Hessenberg Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2d. Generalized Eigenvalue Problems: (Ax = lambda Bx, etc.)</td>
<td></td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>F2c1. Eigenvalues and Eigenvectors of Triangular Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2c2. Eigenvalues and Eigenvectors of Real Symmetric Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2c3. Eigenvalues and Eigenvectors of Real Non-Symmetric Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2c4. Eigenvalues and Eigenvectors of Hessenberg Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2c5. Eigenvalues and Eigenvectors of Complex Hermitian Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2c6. Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2c7. Eigenvalues and Eigenvectors of Sparse Matrices</td>
<td></td>
</tr>
<tr>
<td>Keyword Index to Categories</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Eigenvectors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c8. Eigenvalues and Eigenvectors of Banded Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c1. Eigenvalues and Eigenvectors of Tridiagonal Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c2. Eigenvalues and Eigenvectors of Real Symmetric Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c3. Eigenvalues and Eigenvectors of Real Non-Symmetric Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c4. Eigenvalues and Eigenvectors of Hessenberg Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c5. Eigenvalues and Eigenvectors of Complex Hermitian Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c6. Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c7. Eigenvalues and Eigenvectors of Sparse Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2c8. Eigenvalues and Eigenvectors of Banded Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Elementary</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Arithmetic, Elementary Operations on Polynomials</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1a6. Elementary Rotations (Givens Transformations)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1a7. Elementary Reflections (Householder Transformations)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1a8. Other Elementary Vector Operations, Including Vectorization Aids</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1b8. Other Elementary Matrix Operations</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Elliptic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5m. Elliptic Integrals, Elliptic Functions, Theta Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Equations</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. Roots (Zeros) of Functions, Simultaneous Nonlinear Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2a1. Initial Value Problems for Non-Stiff Ordinary Differential Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2a2. Initial Value Problems for Stiff Ordinary Differential Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2b. Boundary Value Problems for Ordinary Differential Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2c. Eigenvalue Problems for Ordinary Differential Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3. Partial Differential Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5. Integral Equations</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5g. Error Functions, Voigt Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q5. Error Handling Packages</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Experimental</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L07a. Generation of Experimental Designs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exploratory</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L03e. Statistical Graphics: Exploratory Data Analysis Graphics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L16. Exploratory Data Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exponential</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3. Exponential and Logarithmic Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5e. Exponential, Logarithmic, Sine and Cosine Integrals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L06g. Pseudo-Random Number Generators: Exponential</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extremal</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1a1. Vector Norms and Extremal Elements</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Factor</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L13a. Factor Analysis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Keyword Index to Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>G1</td>
<td>Fast Fourier Transforms, Sine and Cosine Transforms</td>
</tr>
<tr>
<td>Fields</td>
<td>J5a3</td>
<td>Graphics: Representing Two-Dimensional Fields</td>
</tr>
<tr>
<td>File</td>
<td>K1</td>
<td>File Management: Access, Deletion, and Staging</td>
</tr>
<tr>
<td></td>
<td>K2</td>
<td>File Copying, Renaming</td>
</tr>
<tr>
<td></td>
<td>K3</td>
<td>File Editing</td>
</tr>
<tr>
<td></td>
<td>K4</td>
<td>File Combination, Splitting, and Conversion Tools</td>
</tr>
<tr>
<td></td>
<td>K5</td>
<td>File Directory, Status, and Size Information</td>
</tr>
<tr>
<td></td>
<td>K6</td>
<td>File Comparison</td>
</tr>
<tr>
<td></td>
<td>K7</td>
<td>File Archiving and Version Control</td>
</tr>
<tr>
<td></td>
<td>K8</td>
<td>File Positioning</td>
</tr>
<tr>
<td></td>
<td>K9</td>
<td>File Characteristics and Permissions (Access Control)</td>
</tr>
<tr>
<td>Filtering</td>
<td>L10a</td>
<td>Time Series: Transformations, Smoothing, Filtering</td>
</tr>
<tr>
<td>Finite</td>
<td>H2a2a</td>
<td>One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a2b</td>
<td>One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3a1</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, General Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3a2</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand and Singularities or Discontinuities</td>
</tr>
<tr>
<td></td>
<td>H2a3b1</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3b2</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand and Singularities or Discontinuities</td>
</tr>
<tr>
<td>Forecasting</td>
<td>L10c</td>
<td>Time Series: ARMA and ARIMA Modeling and Forecasting</td>
</tr>
<tr>
<td>Format</td>
<td>J4</td>
<td>Free Format and Data Structure Input/Output</td>
</tr>
<tr>
<td>Formula</td>
<td>H2c</td>
<td>Computation of Weights and Nodes for Quadrature (Integration) Formula</td>
</tr>
<tr>
<td>Fourier</td>
<td>G1</td>
<td>Fast Fourier Transforms, Sine and Cosine Transforms</td>
</tr>
<tr>
<td>Free</td>
<td>J4</td>
<td>Free Format and Data Structure Input/Output</td>
</tr>
<tr>
<td>Frequencies</td>
<td>L01b</td>
<td>Descriptive Statistics: Distributions, Densities, Frequencies (see also L03a and L03e)</td>
</tr>
<tr>
<td>Full</td>
<td>F4a1</td>
<td>Linear Systems: Real Non-Symmetric Full</td>
</tr>
<tr>
<td></td>
<td>F4b1a</td>
<td>Linear Systems: Real Symmetric Full Indefinite</td>
</tr>
<tr>
<td></td>
<td>F4b1b</td>
<td>Linear Systems: Real Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td></td>
<td>F4c1</td>
<td>Linear Systems: Complex Non-Hermitian Full</td>
</tr>
<tr>
<td></td>
<td>F4d1a</td>
<td>Linear Systems: Complex Hermitian Full Indefinite</td>
</tr>
<tr>
<td></td>
<td>F4d1b</td>
<td>Linear Systems: Complex Hermitian Full Positive-Definite</td>
</tr>
</tbody>
</table>
**F4e1. Linear Systems: Double-Precision Non-Symmetric Full**

**F4f1a. Linear Systems: Double-Precision Symmetric Full Indefinite**

**F4f1b. Linear Systems: Double-Precision Symmetric Full Positive-Definite**

**E1d. Interpolation of a Function of One Variable**

**E1e. Interpolation of a Function of Two or More Variables**

**H1a. Numerical Differentiation: Function Available only on Grid**

**H1b. Numerical Differentiation: User-Defined Function**

**I1a. Optimization (Function Minimization): One-Dimensional**

**I1b. Optimization (Function Minimization): Multi-Dimensional, Unconstrained**

**I2a. Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming**

**I2h1. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints**

**I2h3. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints**

**L05e. Statistical Distributions: Circular Coverage Function**

**L10d. Time Series: Transfer Function Modeling**

**Gamma**

**B5f. Gamma, Beta, Psi Functions**

**L05j. Statistical Distributions: Gamma**

**L06j. Pseudo-Random Number Generators: Gamma**

**Generalized**

**F2d. Generalized Eigenvalue Problems: (Ax = lambda Bx, etc.)**

**F6. Singular Value Decomposition, Rank Determination, Generalized Inverse**

**Generation**

**E9. Mesh Generation for Interpolation or Approximation**

**L07a. Generation of Experimental Designs**

**Generators**

**L06a. Pseudo-Random Number Generators: Beta**

**L06b. Pseudo-Random Number Generators: Binomial**

**L06c. Pseudo-Random Number Generators: Cauchy**

**L06d. Pseudo-Random Number Generators: Chi-squared**

**L06g. Pseudo-Random Number Generators: Exponential**

**L06i. Pseudo-Random Number Generators: F Distribution**

**L06j. Pseudo-Random Number Generators: Gamma**

**L06k. Pseudo-Random Number Generators: Geometric**

**L06m. Pseudo-Random Number Generators: Hypergeometric**

**L06p. Pseudo-Random Number Generators: Logistic**

**L06q. Pseudo-Random Number Generators: Lognormal**

**L06r. Pseudo-Random Number Generators: Negative Binomial**

**L06s. Pseudo-Random Number Generators: Normal**

**L06u. Pseudo-Random Number Generators: Poisson**

**L06v. Pseudo-Random Number Generators: T Distribution**
<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric</td>
<td>L06w. Pseudo-Random Number Generators: Uniform and Associated Uniform Generator Tests</td>
</tr>
<tr>
<td>Givens</td>
<td>F1a6. Elementary Rotations (Givens Transformations)</td>
</tr>
<tr>
<td>Goodness-of-Fit</td>
<td>L04c. Goodness-of-Fit Tests (see also L09)</td>
</tr>
<tr>
<td></td>
<td>J5a2. Graphics: Plotting Graphs and Histograms</td>
</tr>
<tr>
<td></td>
<td>J5a3. Graphics: Representing Two-Dimensional Fields</td>
</tr>
<tr>
<td></td>
<td>J5a4. Graphics: Representing Objects in Three-Space</td>
</tr>
<tr>
<td></td>
<td>J5a5. Graphics: Writing Characters</td>
</tr>
<tr>
<td></td>
<td>J5a7. Graphics: Movie Utilities</td>
</tr>
<tr>
<td></td>
<td>J5b. Graphics: Other Utilities</td>
</tr>
<tr>
<td></td>
<td>L03a. Statistical Graphics: Histograms</td>
</tr>
<tr>
<td></td>
<td>L03b. Statistical Graphics: Scatter Plots (see also J5a2)</td>
</tr>
<tr>
<td></td>
<td>L03c. Statistical Graphics: Symbol Plots</td>
</tr>
<tr>
<td></td>
<td>L03d. Statistical Graphics: Probability Plots</td>
</tr>
<tr>
<td></td>
<td>L03e. Statistical Graphics: Exploratory Data Analysis Graphics</td>
</tr>
<tr>
<td></td>
<td>L03f. Statistical Graphics: Time Series Plots</td>
</tr>
<tr>
<td></td>
<td>L03g. Statistical Graphics: Other Techniques</td>
</tr>
<tr>
<td>Graphs</td>
<td>J5a2. Graphics: Plotting Graphs and Histograms</td>
</tr>
<tr>
<td>Grid</td>
<td>H1a. Numerical Differentiation: Function Available only on Grid</td>
</tr>
<tr>
<td></td>
<td>H2a1. One-Dimensional Integration: Integrand Defined on a Grid</td>
</tr>
<tr>
<td>Hermitian</td>
<td>F2c5. Eigenvalues and Eigenvectors of Complex Hermitian Matrices</td>
</tr>
<tr>
<td></td>
<td>F4d1a. Linear Systems: Complex Hermitian Full Indefinite</td>
</tr>
<tr>
<td></td>
<td>F4d1b. Linear Systems: Complex Hermitian Full Positive-Definite</td>
</tr>
<tr>
<td></td>
<td>F4d2. Linear Systems: Complex Hermitian Banded</td>
</tr>
<tr>
<td>Hessenberg</td>
<td>F2a2. Eigenvalue Problems: Reduction to Hessenberg Form</td>
</tr>
<tr>
<td></td>
<td>F2c4. Eigenvalues and Eigenvectors of Hessenberg Matrices</td>
</tr>
<tr>
<td>Higher</td>
<td>B5z. Miscellaneous Higher Mathematical Functions</td>
</tr>
<tr>
<td>Hilbert</td>
<td>G4. Hilbert Transforms</td>
</tr>
<tr>
<td>Histograms</td>
<td>J5a2. Graphics: Plotting Graphs and Histograms</td>
</tr>
<tr>
<td><strong>Keyword</strong></td>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Householder</td>
<td>F1a7.</td>
</tr>
<tr>
<td>Hyperbolic</td>
<td>B2.</td>
</tr>
<tr>
<td>Hypergeometric</td>
<td>L05m.</td>
</tr>
<tr>
<td></td>
<td>L06m.</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>L04a.</td>
</tr>
<tr>
<td>I</td>
<td>B5il.</td>
</tr>
<tr>
<td>I/O</td>
<td>J1.</td>
</tr>
<tr>
<td></td>
<td>J6.</td>
</tr>
<tr>
<td>Indefinite</td>
<td>F4b1a.</td>
</tr>
<tr>
<td></td>
<td>F4d1a.</td>
</tr>
<tr>
<td></td>
<td>F4f1a.</td>
</tr>
<tr>
<td>Infinite</td>
<td>H2a4.</td>
</tr>
<tr>
<td>Initial</td>
<td>D2a1.</td>
</tr>
<tr>
<td></td>
<td>D2a2.</td>
</tr>
<tr>
<td>Input/Output</td>
<td>J1.</td>
</tr>
<tr>
<td></td>
<td>J2.</td>
</tr>
<tr>
<td></td>
<td>J3.</td>
</tr>
<tr>
<td></td>
<td>J4.</td>
</tr>
<tr>
<td>Integer</td>
<td>M1a1.</td>
</tr>
<tr>
<td>Integral</td>
<td>D5.</td>
</tr>
<tr>
<td>Integrals</td>
<td>B5e.</td>
</tr>
<tr>
<td></td>
<td>B5m.</td>
</tr>
<tr>
<td>Integrand</td>
<td>H2a1.</td>
</tr>
<tr>
<td></td>
<td>H2a2a.</td>
</tr>
<tr>
<td></td>
<td>H2a2b.</td>
</tr>
<tr>
<td></td>
<td>H2a3a1.</td>
</tr>
<tr>
<td></td>
<td>H2a3a2.</td>
</tr>
</tbody>
</table>
# Keyword Index to Categories

## Integration

- **H2a3b1.** One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities
- **H2a3b2.** One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities

## H2a1.** One-Dimensional Integration: Integrand Defined on a Grid
- **H2a2a.** One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand
- **H2a2b.** One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand
- **H2a3a1.** One-Dimensional Integration: Finite Interval, Automatic, General Integrand
- **H2a3a2.** One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities
- **H2a3b1.** One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand
- **H2a3b2.** One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities
- **H2a4.** One-Dimensional Integration: Infinite Interval
- **H2b.** Multi-Dimensional Integration
- **H2c.** Computation of Weights and Nodes for Quadrature (Integration) Formula

## Interpolating

- **E8a.** Evaluation of Interpolating or Approximating Functions using B-Splines
- **E8b.** Evaluation of Interpolating or Approximating Functions using Chebyshev Series
- **E8c.** Evaluation of Interpolating or Approximating Functions using Cubic Splines
- **E8d.** Evaluation of Interpolating or Approximating Functions using Splines under Tension

## Interpolation

- **E1d.** Interpolation of a Function of One Variable
- **E1e.** Interpolation of a Function of Two or More Variables
- **E9.** Mesh Generation for Interpolation or Approximation

## Interval

- **H2a2a.** One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand
- **H2a2b.** One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand
- **H2a3a1.** One-Dimensional Integration: Finite Interval, Automatic, General Integrand
- **H2a3a2.** One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities
- **H2a3b1.** One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand
- **H2a3b2.** One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities
<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>H2a4. One-Dimensional Integration: Infinite Interval</td>
</tr>
<tr>
<td></td>
<td>L04a. Confidence Intervals, Hypothesis Testing</td>
</tr>
<tr>
<td>Inverse</td>
<td>B1. Trigonometric and Inverse Trigonometric Functions</td>
</tr>
<tr>
<td></td>
<td>B2. Hyperbolic and Inverse Hyperbolic Functions</td>
</tr>
<tr>
<td></td>
<td>F6. Singular Value Decomposition, Rank Determination, Generalized Inverse</td>
</tr>
<tr>
<td>J</td>
<td>B5j1. J Bessel Functions of Real Argument</td>
</tr>
<tr>
<td>Job</td>
<td>Q7. Job Control, Job Status, Operating System Requests</td>
</tr>
<tr>
<td>K</td>
<td>B5i2. K Bessel Functions of Real Argument</td>
</tr>
<tr>
<td>Kelvin</td>
<td>B5k. Bessel Functions of Complex Argument, Kelvin Functions</td>
</tr>
<tr>
<td>Kolmogorov-Smirnov</td>
<td>L05n. Statistical Distributions: Kolmogorov-Smirnov</td>
</tr>
<tr>
<td>L-Infinity</td>
<td>E5. Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems</td>
</tr>
<tr>
<td>L04a</td>
<td>L10l. Time Series: Other Time Series Techniques (see also L04a)</td>
</tr>
<tr>
<td>L1</td>
<td>E5. Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems</td>
</tr>
<tr>
<td>Language</td>
<td>S. Software Development Tools, Language Processors</td>
</tr>
<tr>
<td>Laplace</td>
<td>G3. Laplace Transforms</td>
</tr>
<tr>
<td>Least</td>
<td>E2a. Linear Least Squares Approximation, Overdetermined Linear Systems (see also L08a3)</td>
</tr>
<tr>
<td></td>
<td>E2b. Nonlinear Least Squares Approximation (see also L08b)</td>
</tr>
<tr>
<td>Legendre</td>
<td>B5r. Orthogonal Polynomials, Including Legendre Polynomials</td>
</tr>
<tr>
<td>Life</td>
<td>L15. Life Testing, Survival Analysis</td>
</tr>
<tr>
<td>Linear</td>
<td>E2a. Linear Least Squares Approximation, Overdetermined Linear Systems (see also L08a3)</td>
</tr>
<tr>
<td></td>
<td>E5. Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems</td>
</tr>
<tr>
<td></td>
<td>F4a1. Linear Systems: Real Non-Symmetric Full</td>
</tr>
<tr>
<td></td>
<td>F4a2. Linear Systems: Real Non-Symmetric Banded</td>
</tr>
<tr>
<td></td>
<td>F4a4. Linear Systems: Real Non-Symmetric Sparse</td>
</tr>
<tr>
<td></td>
<td>F4b1a. Linear Systems: Real Symmetric Full Indefinite</td>
</tr>
<tr>
<td></td>
<td>F4b1b. Linear Systems: Real Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td></td>
<td>F4b2. Linear Systems: Real Symmetric Banded</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>F4b4.</td>
<td>Linear Systems: Real Symmetric Sparse</td>
</tr>
<tr>
<td>F4c1.</td>
<td>Linear Systems: Complex Non-Hermitian Full</td>
</tr>
<tr>
<td>F4c2.</td>
<td>Linear Systems: Complex Non-Hermitian Banded</td>
</tr>
<tr>
<td>F4d1a.</td>
<td>Linear Systems: Complex Hermitian Full Indefinite</td>
</tr>
<tr>
<td>F4d1b.</td>
<td>Linear Systems: Complex Hermitian Full Positive-Definite</td>
</tr>
<tr>
<td>F4d2.</td>
<td>Linear Systems: Complex Hermitian Banded</td>
</tr>
<tr>
<td>F4e1.</td>
<td>Linear Systems: Double-Precision Non-Symmetric Full</td>
</tr>
<tr>
<td>F4e2.</td>
<td>Linear Systems: Double-Precision Non-Symmetric Banded</td>
</tr>
<tr>
<td>F4f1a.</td>
<td>Linear Systems: Double-Precision Symmetric Full Indefinite</td>
</tr>
<tr>
<td>F4f1b.</td>
<td>Linear Systems: Double-Precision Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td>F4f2.</td>
<td>Linear Systems: Double-Precision Symmetric Banded</td>
</tr>
<tr>
<td>I2a.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming</td>
</tr>
<tr>
<td>I2h1.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
<tr>
<td>L08a1.</td>
<td>Regression: Simple Linear</td>
</tr>
<tr>
<td>L08a3.</td>
<td>Regression: Multiple Linear</td>
</tr>
<tr>
<td>J5a1.</td>
<td>Graphics: Plotting Dashed Lines</td>
</tr>
<tr>
<td>L01a.</td>
<td>Descriptive Statistics: Location, Dispersion, Shape</td>
</tr>
<tr>
<td>B3.</td>
<td>Exponential and Logarithmic Functions</td>
</tr>
<tr>
<td>B5e.</td>
<td>Exponential, Logarithmic, Sine and Cosine Integrals</td>
</tr>
<tr>
<td>L06p.</td>
<td>Pseudo-Random Number Generators: Logistic</td>
</tr>
<tr>
<td>L06q.</td>
<td>Pseudo-Random Number Generators: Lognormal</td>
</tr>
<tr>
<td>Q3.</td>
<td>Machine and System Dependent Constants</td>
</tr>
<tr>
<td>K1.</td>
<td>File Management: Access, Deletion, and Staging</td>
</tr>
<tr>
<td>Q8.</td>
<td>Storage Allocation, Memory Management, and Overlays</td>
</tr>
<tr>
<td>L02.</td>
<td>Data Manipulation</td>
</tr>
<tr>
<td>M4.</td>
<td>Character Manipulation, Bit Operations</td>
</tr>
<tr>
<td>D7.</td>
<td>Conformal Mapping, Schwarz-Christoffel Transformations</td>
</tr>
<tr>
<td>F2c1.</td>
<td>Eigenvalues and Eigenvectors of Tridiagonal Matrices</td>
</tr>
<tr>
<td>F2c2.</td>
<td>Eigenvalues and Eigenvectors of Real Symmetric Matrices</td>
</tr>
<tr>
<td>F2c3.</td>
<td>Eigenvalues and Eigenvectors of Real Non-Symmetric Matrices</td>
</tr>
<tr>
<td>F2c4.</td>
<td>Eigenvalues and Eigenvectors of Hessenberg Matrices</td>
</tr>
<tr>
<td>F2c5.</td>
<td>Eigenvalues and Eigenvectors of Complex Hermitian Matrices</td>
</tr>
<tr>
<td>F2c6.</td>
<td>Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices</td>
</tr>
<tr>
<td>Category</td>
<td>Keyword Index to Categories</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Matrices</td>
<td>F2c7. Eigenvalues and Eigenvectors of Sparse Matrices</td>
</tr>
<tr>
<td></td>
<td>F2c8. Eigenvalues and Eigenvectors of Banded Matrices</td>
</tr>
<tr>
<td>Matrix</td>
<td>F1b3. Matrix Multiplication</td>
</tr>
<tr>
<td></td>
<td>F1b7. Matrix Storage Mode Conversion</td>
</tr>
<tr>
<td></td>
<td>F1b8. Other Elementary Matrix Operations</td>
</tr>
<tr>
<td></td>
<td>F7. Updating Matrix Decompositions</td>
</tr>
<tr>
<td>Memory</td>
<td>Q6. Storage Allocation, Memory Management, and Overlays</td>
</tr>
<tr>
<td>Mesh</td>
<td>E9. Mesh Generation for Interpolation or Approximation</td>
</tr>
<tr>
<td>Minimax</td>
<td>E5. Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems</td>
</tr>
<tr>
<td>Minimization</td>
<td>I1a. Optimization (Function Minimization): One-Dimensional</td>
</tr>
<tr>
<td></td>
<td>I1b. Optimization (Function Minimization): Multi-Dimensional, Unconstrained</td>
</tr>
<tr>
<td></td>
<td>I2a. Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming</td>
</tr>
<tr>
<td></td>
<td>I2h1. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
<tr>
<td></td>
<td>I2h3. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>B5l. Miscellaneous Bessel and Related Functions, Airy Functions</td>
</tr>
<tr>
<td></td>
<td>B5z. Miscellaneous Higher Mathematical Functions</td>
</tr>
<tr>
<td></td>
<td>J6. Miscellaneous I/O, Including Communications, Special Devices</td>
</tr>
<tr>
<td>Missing</td>
<td>L11a. Correlation Analysis: No Missing Values</td>
</tr>
<tr>
<td></td>
<td>L11b. Correlation Analysis: Missing Values</td>
</tr>
<tr>
<td>Mode</td>
<td>F1b7. Matrix Storage Mode Conversion</td>
</tr>
<tr>
<td>Modeling</td>
<td>L10c. Time Series: ARMA and ARIMA Modeling and Forecasting</td>
</tr>
<tr>
<td></td>
<td>L10d. Time Series: Transfer Function Modeling</td>
</tr>
<tr>
<td>Movie</td>
<td>J5a7. Graphics: Movie Utilities</td>
</tr>
<tr>
<td>Multi-Dimensional</td>
<td>H2b. Multi-Dimensional Integration</td>
</tr>
<tr>
<td></td>
<td>I1b. Optimization (Function Minimization): Multi-Dimensional, Unconstrained</td>
</tr>
<tr>
<td></td>
<td>I2a. Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming</td>
</tr>
<tr>
<td></td>
<td>I2h1. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
</tbody>
</table>
Keyword Index to Categories

I2h3. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints

Multiple
L08a3. Regression: Multiple Linear

Multiplication
F1b3. Matrix Multiplication

Multivariate
L07e. Multivariate Analysis of Variance
L10b. Time Series: Autocorrelation and Cross-Correlation, Univariate and Multivariate
L10f. Time Series: Multivariate Spectral Analysis

Negative
L05r. Statistical Distributions: Negative Binomial
L06r. Pseudo-Random Number Generators: Negative Binomial

Nodes
H2c. Computation of Weights and Nodes for Quadrature (Integration) Formula

Non-Hermitian
F2c6. Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices
F4c1. Linear Systems: Complex Non-Hermitian Full
F4c2. Linear Systems: Complex Non-Hermitian Banded

Non-Stiff
D2a1. Initial Value Problems for Non-Stiff Ordinary Differential Equations

Non-Symmetric
F2c3. Eigenvalues and Eigenvectors of Real Non-Symmetric Matrices
F4a1. Linear Systems: Real Non-Symmetric Full
F4a2. Linear Systems: Real Non-Symmetric Banded
F4a4. Linear Systems: Real Non-Symmetric Sparse
F4e1. Linear Systems: Double-Precision Non-Symmetric Full
F4e2. Linear Systems: Double-Precision Non-Symmetric Banded

Non-User-Callable
Z. Non-User-Callable Support Software

Nonautomatic
H2a2a. One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand
H2a2b. One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand

Nonlinear
C. Roots (Zeros) of Functions, Simultaneous Nonlinear Equations
E2b. Nonlinear Least Squares Approximation (see also L08b)
I2h1. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints
I2h3. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints
L08b. Regression: Nonlinear
<table>
<thead>
<tr>
<th>Category</th>
<th>Keyword Index to Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonparametric</td>
<td>L04b. Nonparametric (Distribution-Free) Analysis (see also L07c, L11a, L11b)</td>
</tr>
<tr>
<td></td>
<td>L07c. Nonparametric Analysis of Variance</td>
</tr>
<tr>
<td>Normal</td>
<td>L05s. Statistical Distributions: Normal</td>
</tr>
<tr>
<td></td>
<td>L06s. Pseudo-Random Number Generators: Normal</td>
</tr>
<tr>
<td>Norms</td>
<td>F1a1. Vector Norms and Extremal Elements</td>
</tr>
<tr>
<td>Numerical</td>
<td>H1a. Numerical Differentiation: Function Available only on Grid</td>
</tr>
<tr>
<td></td>
<td>H1b. Numerical Differentiation: User-Defined Function</td>
</tr>
<tr>
<td>Objects</td>
<td>J5a4. Graphics: Representing Objects in Three-Space</td>
</tr>
<tr>
<td>One-Dimensional</td>
<td>H2a1. One-Dimensional Integration: Integrand Defined on a Grid</td>
</tr>
<tr>
<td></td>
<td>H2a2a. One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a2b. One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3a1. One-Dimensional Integration: Finite Interval, Automatic, General Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3a2. One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td></td>
<td>H2a3b1. One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3b2. One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td></td>
<td>H2a4. One-Dimensional Integration: Infinite Interval</td>
</tr>
<tr>
<td></td>
<td>I1a. Optimization (Function Minimization): One-Dimensional</td>
</tr>
<tr>
<td>Operating</td>
<td>Q7. Job Control, Job Status, Operating System Requests</td>
</tr>
<tr>
<td>Operations</td>
<td>A. Arithmetic, Elementary Operations on Polynomials</td>
</tr>
<tr>
<td></td>
<td>F1a8. Other Elementary Vector Operations, Including Vectorization Aids</td>
</tr>
<tr>
<td></td>
<td>F1b8. Other Elementary Matrix Operations</td>
</tr>
<tr>
<td></td>
<td>M4. Character Manipulation, Bit Operations</td>
</tr>
<tr>
<td>Optimization</td>
<td>I1a. Optimization (Function Minimization): One-Dimensional</td>
</tr>
<tr>
<td></td>
<td>I1b. Optimization (Function Minimization): Multi-Dimensional, Unconstrained</td>
</tr>
<tr>
<td></td>
<td>I2a. Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming</td>
</tr>
<tr>
<td></td>
<td>I2h1. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
<tr>
<td></td>
<td>I2h3. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints</td>
</tr>
<tr>
<td>Category</td>
<td>Keyword Index to Categories</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Ordinary</td>
<td>D2a1. Initial Value Problems for Non-Stiff Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D2a2. Initial Value Problems for Stiff Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D2b. Boundary Value Problems for Ordinary Differential Equations</td>
</tr>
<tr>
<td></td>
<td>D2c. Eigenvalue Problems for Ordinary Differential Equations</td>
</tr>
<tr>
<td>Orthogonal</td>
<td>B5r. Orthogonal Polynomials, Including Legendre Polynomials</td>
</tr>
<tr>
<td></td>
<td>F5. QR (Orthogonal) Decomposition</td>
</tr>
<tr>
<td>Overdetermined</td>
<td>E2a. Linear Least Squares Approximation, Overdetermined Linear Systems (see also L08a3)</td>
</tr>
<tr>
<td></td>
<td>E5. Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems</td>
</tr>
<tr>
<td>Overlays</td>
<td>Q6. Storage Allocation, Memory Management, and Overlays</td>
</tr>
<tr>
<td>Packing</td>
<td>M2. Conversion, Packing and Unpacking</td>
</tr>
<tr>
<td>Partial</td>
<td>D3. Partial Differential Equations</td>
</tr>
<tr>
<td>Permissions</td>
<td>K9. File Characteristics and Permissions (Access Control)</td>
</tr>
<tr>
<td>Plots</td>
<td>L03b. Statistical Graphics: Scatter Plots (see also J5a2)</td>
</tr>
<tr>
<td></td>
<td>L03c. Statistical Graphics: Symbol Plots</td>
</tr>
<tr>
<td></td>
<td>L03d. Statistical Graphics: Probability Plots</td>
</tr>
<tr>
<td></td>
<td>L03f. Statistical Graphics: Time Series Plots</td>
</tr>
<tr>
<td>Plotting</td>
<td>J5a1. Graphics: Plotting Dashed Lines</td>
</tr>
<tr>
<td></td>
<td>J5a2. Graphics: Plotting Graphs and Histograms</td>
</tr>
<tr>
<td>Poisson</td>
<td>L05u. Statistical Distributions: Poisson</td>
</tr>
<tr>
<td></td>
<td>L06u. Pseudo-Random Number Generators: Poisson</td>
</tr>
<tr>
<td>Polynomial</td>
<td>L08a2. Regression: Polynomial</td>
</tr>
<tr>
<td>Polynomials</td>
<td>A. Arithmetic, Elementary Operations on Polynomials</td>
</tr>
<tr>
<td></td>
<td>B5r. Orthogonal Polynomials, Including Legendre Polynomials</td>
</tr>
<tr>
<td>Positioning</td>
<td>K8. File Positioning</td>
</tr>
<tr>
<td>Positive-Definite</td>
<td>F4b1b. Linear Systems: Real Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td></td>
<td>F4d1b. Linear Systems: Complex Hermitian Full Positive-Definite</td>
</tr>
<tr>
<td></td>
<td>F4f1b. Linear Systems: Double-Precision Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td>Principal</td>
<td>L13b. Principal Component Analysis</td>
</tr>
<tr>
<td>Probability</td>
<td>L03d. Statistical Graphics: Probability Plots</td>
</tr>
<tr>
<td>Processors</td>
<td>S.</td>
</tr>
<tr>
<td>Product</td>
<td>F1a2.</td>
</tr>
<tr>
<td>Programming</td>
<td>I2a.</td>
</tr>
<tr>
<td>Pseudo-Random</td>
<td>L06a.</td>
</tr>
<tr>
<td></td>
<td>L06b.</td>
</tr>
<tr>
<td></td>
<td>L06c.</td>
</tr>
<tr>
<td></td>
<td>L06d.</td>
</tr>
<tr>
<td></td>
<td>L06g.</td>
</tr>
<tr>
<td></td>
<td>L06i.</td>
</tr>
<tr>
<td></td>
<td>L06j.</td>
</tr>
<tr>
<td></td>
<td>L06k.</td>
</tr>
<tr>
<td></td>
<td>L06m.</td>
</tr>
<tr>
<td></td>
<td>L06p.</td>
</tr>
<tr>
<td></td>
<td>L06q.</td>
</tr>
<tr>
<td></td>
<td>L06r.</td>
</tr>
<tr>
<td></td>
<td>L06s.</td>
</tr>
<tr>
<td></td>
<td>L06u.</td>
</tr>
<tr>
<td></td>
<td>L06v.</td>
</tr>
<tr>
<td></td>
<td>L06w.</td>
</tr>
<tr>
<td></td>
<td>L06x.</td>
</tr>
<tr>
<td></td>
<td>L06y.</td>
</tr>
<tr>
<td></td>
<td>L06z.</td>
</tr>
<tr>
<td>Psi</td>
<td>B5f.</td>
</tr>
<tr>
<td>QR</td>
<td>F5.</td>
</tr>
<tr>
<td>Quadratic</td>
<td>I2a.</td>
</tr>
<tr>
<td>Quadrature</td>
<td>H2c.</td>
</tr>
<tr>
<td>Random</td>
<td>J2.</td>
</tr>
<tr>
<td>Rank</td>
<td>F6.</td>
</tr>
<tr>
<td>Real</td>
<td>B5i1.</td>
</tr>
<tr>
<td></td>
<td>B5i2.</td>
</tr>
<tr>
<td></td>
<td>B5j1.</td>
</tr>
<tr>
<td></td>
<td>B5j2.</td>
</tr>
<tr>
<td></td>
<td>F2c2.</td>
</tr>
<tr>
<td></td>
<td>F2c3.</td>
</tr>
</tbody>
</table>
Keyword Index to Categories

Matrices
F4a1. Linear Systems: Real Non-Symmetric Full
F4a2. Linear Systems: Real Non-Symmetric Banded
F4a4. Linear Systems: Real Non-Symmetric Sparse
F4b1a. Linear Systems: Real Symmetric Full Indefinite
F4b1b. Linear Systems: Real Symmetric Full Positive-Definite
F4b2. Linear Systems: Real Symmetric Banded
F4b4. Linear Systems: Real Symmetric Sparse
M1a2. Sorting of Real Data

Reduction
F2a1. Eigenvalue Problems: Reduction to Symmetric Tridiagonal Form
F2a2. Eigenvalue Problems: Reduction to Hessenberg Form

Reflections
F1a7. Elementary Reflections (Householder Transformations)

Regression
L08a1. Regression: Simple Linear
L08a2. Regression: Polynomial
L08a3. Regression: Multiple Linear
L08a4. Regression: Robust
L08b. Regression: Nonlinear
L08c. Regression: Other
L08d. Techniques Used with Regression

Renaming
K2. File Copying, Renaming

Retrieval
Q4. Documentation Retrieval

Robust
L08a4. Regression: Robust

Roots
C. Roots (Zeros) of Functions, Simultaneous Nonlinear Equations

Rotations
F1a6. Elementary Rotations (Givens Transformations)

Sampling
L17. Sampling Techniques

Scaling
F1a5. Vector Scaling ($v1 = s*v1 + v2$)

Scatter
L03b. Statistical Graphics: Scatter Plots (see also J5a2)

Schwarz-Christoffel
D7. Conformal Mapping, Schwarz-Christoffel Transformations

Searching
M5. Searching

Sequential
J1. Sequential Input/Output, Including Tape I/O

Shape
L01a. Descriptive Statistics: Location, Dispersion, Shape

Simultaneous
C. Roots (Zeros) of Functions, Simultaneous Nonlinear
<table>
<thead>
<tr>
<th>Keyword Index to Categories</th>
<th>NCAR Software Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine</td>
<td>B5e.</td>
</tr>
<tr>
<td></td>
<td>Exponential, Logarithmic, Sine and Cosine Integrals</td>
</tr>
<tr>
<td></td>
<td>G1.</td>
</tr>
<tr>
<td></td>
<td>Fast Fourier Transforms, Sine and Cosine Transforms</td>
</tr>
<tr>
<td>Singular</td>
<td>F6.</td>
</tr>
<tr>
<td></td>
<td>Singular Value Decomposition, Rank Determination, Generalized Inverse</td>
</tr>
<tr>
<td>Singularities</td>
<td>H2a3a2.</td>
</tr>
<tr>
<td></td>
<td>One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td></td>
<td>H2a3b2.</td>
</tr>
<tr>
<td></td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td>Size</td>
<td>K5.</td>
</tr>
<tr>
<td></td>
<td>File Directory, Status, and Size Information</td>
</tr>
<tr>
<td>Smoothing</td>
<td>L10a.</td>
</tr>
<tr>
<td></td>
<td>Time Series: Transformations, Smoothing, Filtering</td>
</tr>
<tr>
<td>Sorting</td>
<td>M1a1.</td>
</tr>
<tr>
<td></td>
<td>Sorting of Integer Data</td>
</tr>
<tr>
<td></td>
<td>M1a2.</td>
</tr>
<tr>
<td></td>
<td>Sorting of Real Data</td>
</tr>
<tr>
<td></td>
<td>M1a3.</td>
</tr>
<tr>
<td></td>
<td>Sorting of Character Data</td>
</tr>
<tr>
<td>Sparse</td>
<td>F2c7.</td>
</tr>
<tr>
<td></td>
<td>Eigenvalues and Eigenvectors of Sparse Matrices</td>
</tr>
<tr>
<td></td>
<td>F4a4.</td>
</tr>
<tr>
<td></td>
<td>Linear Systems: Real Non-Symmetric Sparse</td>
</tr>
<tr>
<td></td>
<td>F4b4.</td>
</tr>
<tr>
<td></td>
<td>Linear Systems: Real Symmetric Sparse</td>
</tr>
<tr>
<td>Spectral</td>
<td>L10e.</td>
</tr>
<tr>
<td></td>
<td>Time Series: Univariate Spectral Analysis</td>
</tr>
<tr>
<td></td>
<td>L10f.</td>
</tr>
<tr>
<td></td>
<td>Time Series: Multivariate Spectral Analysis</td>
</tr>
<tr>
<td>Splines</td>
<td>E8c.</td>
</tr>
<tr>
<td></td>
<td>Evaluation of Interpolating or Approximating Functions using Cubic Splines</td>
</tr>
<tr>
<td></td>
<td>E8d.</td>
</tr>
<tr>
<td></td>
<td>Evaluation of Interpolating or Approximating Functions using Splines under Tension</td>
</tr>
<tr>
<td>Splitting</td>
<td>K4.</td>
</tr>
<tr>
<td></td>
<td>File Combination, Splitting, and Conversion Tools</td>
</tr>
<tr>
<td>Staging</td>
<td>K1.</td>
</tr>
<tr>
<td></td>
<td>File Management: Access, Deletion, and Staging</td>
</tr>
<tr>
<td>Status</td>
<td>K5.</td>
</tr>
<tr>
<td></td>
<td>File Directory, Status, and Size Information</td>
</tr>
<tr>
<td></td>
<td>Q7.</td>
</tr>
<tr>
<td></td>
<td>Job Control, Job Status, Operating System Requests</td>
</tr>
<tr>
<td>Stiff</td>
<td>D2a2.</td>
</tr>
<tr>
<td></td>
<td>Initial Value Problems for Stiff Ordinary Differential Equations</td>
</tr>
<tr>
<td>Storage</td>
<td>F1b7.</td>
</tr>
<tr>
<td></td>
<td>Matrix Storage Mode Conversion</td>
</tr>
<tr>
<td></td>
<td>Q6.</td>
</tr>
<tr>
<td></td>
<td>Storage Allocation, Memory Management, and Overlays</td>
</tr>
<tr>
<td>Structure</td>
<td>J4.</td>
</tr>
<tr>
<td></td>
<td>Free Format and Data Structure Input/Output</td>
</tr>
<tr>
<td>Support</td>
<td>Z.</td>
</tr>
<tr>
<td></td>
<td>Non-User-Callable Support Software</td>
</tr>
<tr>
<td>Category</td>
<td>Code</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>Survival</td>
<td>L15</td>
</tr>
<tr>
<td>Swap</td>
<td>F1a3</td>
</tr>
<tr>
<td>Symbol</td>
<td>L03c</td>
</tr>
<tr>
<td>Symmetric</td>
<td>F2a1</td>
</tr>
<tr>
<td></td>
<td>F2c2</td>
</tr>
<tr>
<td></td>
<td>F4b1a</td>
</tr>
<tr>
<td></td>
<td>F4b1b</td>
</tr>
<tr>
<td></td>
<td>F4b2</td>
</tr>
<tr>
<td></td>
<td>F4b4</td>
</tr>
<tr>
<td></td>
<td>F4f1a</td>
</tr>
<tr>
<td></td>
<td>F4f1b</td>
</tr>
<tr>
<td></td>
<td>F4f2</td>
</tr>
<tr>
<td>System</td>
<td>Q3</td>
</tr>
<tr>
<td></td>
<td>Q7</td>
</tr>
<tr>
<td>Systems</td>
<td>E2a</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E5</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F4a1</td>
</tr>
<tr>
<td></td>
<td>F4a2</td>
</tr>
<tr>
<td></td>
<td>F4a4</td>
</tr>
<tr>
<td></td>
<td>F4b1a</td>
</tr>
<tr>
<td></td>
<td>F4b1b</td>
</tr>
<tr>
<td></td>
<td>F4b2</td>
</tr>
<tr>
<td></td>
<td>F4b4</td>
</tr>
<tr>
<td></td>
<td>F4c1</td>
</tr>
<tr>
<td></td>
<td>F4c2</td>
</tr>
<tr>
<td></td>
<td>F4d1a</td>
</tr>
<tr>
<td></td>
<td>F4d1b</td>
</tr>
<tr>
<td></td>
<td>F4d2</td>
</tr>
<tr>
<td></td>
<td>F4e1</td>
</tr>
<tr>
<td></td>
<td>F4e2</td>
</tr>
<tr>
<td></td>
<td>F4f1a</td>
</tr>
<tr>
<td></td>
<td>F4f1b</td>
</tr>
<tr>
<td></td>
<td>F4f2</td>
</tr>
<tr>
<td>T</td>
<td>L05v</td>
</tr>
<tr>
<td></td>
<td>L06v</td>
</tr>
<tr>
<td>Tape</td>
<td>J1</td>
</tr>
</tbody>
</table>
Tension  E8d.  Evaluation of Interpolating or Approximating Functions using Splines under Tension

Testing  L04a.  Confidence Intervals, Hypothesis Testing
           L15.  Life Testing, Survival Analysis

Tests  L04c.  Goodness-of-Fit Tests (see also L09)
       L06w.  Pseudo-Random Number Generators: Uniform and Associated Uniform Generator Tests
       L07f.  Tests to use with Analysis of Variance and Covariance

Theta  B5m.  Elliptic Integrals, Elliptic Functions, Theta Functions

Three-Space  J5a4.  Graphics: Representing Objects in Three-Space

Time  L03f.  Statistical Graphics: Time Series Plots
       L10a.  Time Series: Transformations, Smoothing, Filtering
       L10b.  Time Series: Autocorrelation and Cross-Correlation, Univariate and Multivariate
       L10c.  Time Series: ARMA and ARIMA Modeling and Forecasting
       L10d.  Time Series: Transfer Function Modeling
       L10e.  Time Series: Univariate Spectral Analysis
       L10f.  Time Series: Multivariate Spectral Analysis
       L10l.  Time Series: Other Time Series Techniques (see also L04a)
       Q1.  Date and Time, Program Timers

Timers  Q1.  Date and Time, Program Timers

Tools  K4.  File Combination, Splitting, and Conversion Tools
       S.  Software Development Tools, Language Processors

Transfer  L10d.  Time Series: Transfer Function Modeling

Transformations  D7.  Conformal Mapping, Schwarz-Christoffel Transformations
                 F1a6.  Elementary Rotations (Givens Transformations)
                 F1a7.  Elementary Reflections (Householder Transformations)
                 L10a.  Time Series: Transformations, Smoothing, Filtering

Transforms  G1.  Fast Fourier Transforms, Sine and Cosine Transforms
            G3.  Laplace Transforms
            G4.  Hilbert Transforms

Tridiagonal  F2a1.  Eigenvalue Problems: Reduction to Symmetric Tridiagonal Form
            F2c1.  Eigenvalues and Eigenvectors of Tridiagonal Matrices

Trigonometric  B1.  Trigonometric and Inverse Trigonometric Functions
<table>
<thead>
<tr>
<th>Category</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Dimensional</td>
<td>J5a3</td>
<td>Graphics: Representing Two-Dimensional Fields</td>
</tr>
<tr>
<td>Unconstrained</td>
<td>I1b</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Unconstrained</td>
</tr>
<tr>
<td>Uniform</td>
<td>L06w</td>
<td>Pseudo-Random Number Generators: Uniform and Associated Uniform Generator Tests</td>
</tr>
<tr>
<td>Unpacking</td>
<td>M2</td>
<td>Conversion, Packing and Unpacking</td>
</tr>
<tr>
<td>Updating</td>
<td>F7</td>
<td>Updating Matrix Decompositions</td>
</tr>
<tr>
<td>User-Defined</td>
<td>H1b</td>
<td>Numerical Differentiation: User-Defined Function</td>
</tr>
<tr>
<td>Utilities</td>
<td>J5a7</td>
<td>Graphics: Movie Utilities</td>
</tr>
<tr>
<td></td>
<td>J5b</td>
<td>Graphics: Other Utilities</td>
</tr>
<tr>
<td>Values</td>
<td>L11a</td>
<td>Correlation Analysis: No Missing Values</td>
</tr>
<tr>
<td></td>
<td>L11b</td>
<td>Correlation Analysis: Missing Values</td>
</tr>
<tr>
<td>Variable</td>
<td>E1d</td>
<td>Interpolation of a Function of One Variable</td>
</tr>
<tr>
<td>Variables</td>
<td>E1e</td>
<td>Interpolation of a Function of Two or More Variables</td>
</tr>
<tr>
<td>Variance</td>
<td>L07b</td>
<td>Analysis of Variance</td>
</tr>
<tr>
<td></td>
<td>L07c</td>
<td>Nonparametric Analysis of Variance</td>
</tr>
<tr>
<td></td>
<td>L07e</td>
<td>Multivariate Analysis of Variance</td>
</tr>
<tr>
<td></td>
<td>L07f</td>
<td>Tests to use with Analysis of Variance and Covariance</td>
</tr>
<tr>
<td>Vector</td>
<td>F1a1</td>
<td>Vector Norms and Extremal Elements</td>
</tr>
<tr>
<td></td>
<td>F1a2</td>
<td>Vector Dot Product</td>
</tr>
<tr>
<td></td>
<td>F1a3</td>
<td>Vector Copy or Swap</td>
</tr>
<tr>
<td></td>
<td>F1a5</td>
<td>Vector Scaling ($v1 = s*v1 + v2$)</td>
</tr>
<tr>
<td></td>
<td>F1a8</td>
<td>Other Elementary Vector Operations, Including Vectorization Aids</td>
</tr>
<tr>
<td>Vectorization</td>
<td>F1a8</td>
<td>Other Elementary Vector Operations, Including Vectorization Aids</td>
</tr>
<tr>
<td>Version</td>
<td>K7</td>
<td>File Archiving and Version Control</td>
</tr>
<tr>
<td>Voigt</td>
<td>B5g</td>
<td>Error Functions, Voigt Functions</td>
</tr>
<tr>
<td>Weibull</td>
<td>L06x</td>
<td>Pseudo-Random Number Generators: Weibull</td>
</tr>
<tr>
<td>Weighted</td>
<td>H2a2b</td>
<td>One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3b1</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand</td>
</tr>
<tr>
<td></td>
<td>H2a3b2</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand</td>
</tr>
<tr>
<td>Category</td>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Weights</td>
<td>H²c.</td>
<td>Computation of Weights and Nodes for Quadrature (Integration) Formula</td>
</tr>
<tr>
<td>Writing</td>
<td>J⁵a⁵.</td>
<td>Graphics: Writing Characters</td>
</tr>
<tr>
<td>Y</td>
<td>B⁵j².</td>
<td>Y Bessel Functions of Real Argument</td>
</tr>
<tr>
<td>Zeros</td>
<td>C.</td>
<td>Roots (Zeros) of Functions, Simultaneous Nonlinear Equations</td>
</tr>
</tbody>
</table>
SOFTWARE CATEGORIES

Introduction

This section presents the NCAR software classification scheme. Users will probably be able to locate a software category most easily by using the keyword index of categories in the preceding section, but understanding the organization of the categories may help in finding the category for an area of software that does not have any obvious keywords. Here related categories are generally close together, so selecting an exactly appropriate category from several closely related categories may be easier than with the keyword index.

Top-Level Categories

The first letter of each category identifies which broad area of software it falls under. Some of these broad areas are also actual categories (A, C, N, and S) but most are further subdivided into smaller subcategories. Here are the broad areas covered by this catalog:

A. Arithmetic, Elementary Operations on Polynomials
B. Elementary and Special Functions
C. Roots (Zeros) of Functions, Simultaneous Nonlinear Equations
D. Differential and Integral Equations
E. Interpolation and Approximation
F. Linear Algebra
G. Integral Transforms (including Fast Fourier Transforms)
H. Differentiation, Integration
I. Optimization: Minimizing or Maximizing a Function
J. Input/Output, Graphics
K. File Management
L. Statistics and Probability
M. Data Handling (Sorting, Searching, Character and Bit Operations, Packing)
N. Debugging
Q. System Utilities
S. Software Development Tools, Language Processors
Z. Non-User-Callable Support Software

The NCAR software classification scheme is based on a hierarchy of categories adapted to NCAR’s existing public software collection. To facilitate locating software with this classification scheme, only the “leaves” of the classification tree are used. Furthermore, there are no empty categories: categories for which no software exists at NCAR do not appear in this list. A category may be subdivided into finer subcategories when too many software packages appear within it.
A. Arithmetic, Elementary Operations on Polynomials

B1. Trigonometric and Inverse Trigonometric Functions
B2. Hyperbolic and Inverse Hyperbolic Functions
B3. Exponential and Logarithmic Functions
B5e. Exponential, Logarithmic, Sine and Cosine Integrals
B5f. Gamma, Beta, Psi Functions
B5g. Error Functions, Voigt Functions
B5i. I Bessel Functions of Real Argument
B5j. K Bessel Functions of Real Argument
B5k. J Bessel Functions of Real Argument
B5l. Y Bessel Functions of Real Argument
B5m. Bessel Functions of Complex Argument, Kelvin Functions
B5n. Miscellaneous Bessel and Related Functions, Airy Functions
B5o. Elliptic Integrals, Elliptic Functions, Theta Functions
B5p. Orthogonal Polynomials, Including Legendre Polynomials
B5q. Miscellaneous Higher Mathematical Functions

C. Roots (Zeros) of Functions, Simultaneous Nonlinear Equations

D2a1. Initial Value Problems for Non-Stiff Ordinary Differential Equations
D2a2. Initial Value Problems for Stiff Ordinary Differential Equations
D2b. Boundary Value Problems for Ordinary Differential Equations
D2c. Eigenvalue Problems for Ordinary Differential Equations
D3. Partial Differential Equations
D5. Integral Equations
D7. Conformal Mapping, Schwarz-Christoffel Transformations

E1d. Interpolation of a Function of One Variable
E1e. Interpolation of a Function of Two or More Variables
E2a. Linear Least Squares Approximation, Overdetermined Linear Systems (see also L08a3)
E2b. Nonlinear Least Squares Approximation (see also L08b)
E5. Minimax (L-Infinity), L1, and Other Approximations, Overdetermined Linear Systems
E8a. Evaluation of Interpolating or Approximating Functions using B-Splines
E8b. Evaluation of Interpolating or Approximating Functions using Chebyshev Series
E8c. Evaluation of Interpolating or Approximating Functions using Cubic Splines
E8d. Evaluation of Interpolating or Approximating Functions using Splines under Tension
E9. Mesh Generation for Interpolation or Approximation

F1a1. Vector Norms and Extremal Elements
F1a2. Vector Dot Product
F1a3. Vector Copy or Swap
F1a5. Vector Scaling ($v1 = s*v1 + v2$)
F1a6. Elementary Rotations (Givens Transformations)
<table>
<thead>
<tr>
<th>Software Categories</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1a7.</td>
<td>Elementary Reflections (Householder Transformations)</td>
</tr>
<tr>
<td>F1a8.</td>
<td>Other Elementary Vector Operations, Including Vectorization Aids</td>
</tr>
<tr>
<td>F1b3.</td>
<td>Matrix Multiplication</td>
</tr>
<tr>
<td>F1b7.</td>
<td>Matrix Storage Mode Conversion</td>
</tr>
<tr>
<td>F1b8.</td>
<td>Other Elementary Matrix Operations</td>
</tr>
<tr>
<td>F2a1.</td>
<td>Eigenvalue Problems: Reduction to Symmetric Tridiagonal Form</td>
</tr>
<tr>
<td>F2a2.</td>
<td>Eigenvalue Problems: Reduction to Hessenberg Form</td>
</tr>
<tr>
<td>F2c1.</td>
<td>Eigenvalues and Eigenvectors of Tridiagonal Matrices</td>
</tr>
<tr>
<td>F2c2.</td>
<td>Eigenvalues and Eigenvectors of Real Symmetric Matrices</td>
</tr>
<tr>
<td>F2c3.</td>
<td>Eigenvalues and Eigenvectors of Real Non-Symmetric Matrices</td>
</tr>
<tr>
<td>F2c4.</td>
<td>Eigenvalues and Eigenvectors of Hessenberg Matrices</td>
</tr>
<tr>
<td>F2c5.</td>
<td>Eigenvalues and Eigenvectors of Complex Hermitian Matrices</td>
</tr>
<tr>
<td>F2c6.</td>
<td>Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices</td>
</tr>
<tr>
<td>F2c7.</td>
<td>Eigenvalues and Eigenvectors of Sparse Matrices</td>
</tr>
<tr>
<td>F2c8.</td>
<td>Eigenvalues and Eigenvectors of Banded Matrices</td>
</tr>
<tr>
<td>F2d.</td>
<td>Generalized Eigenvalue Problems: (Ax = lambda Bx, etc.)</td>
</tr>
<tr>
<td>F4a1.</td>
<td>Linear Systems: Real Non-Symmetric Full</td>
</tr>
<tr>
<td>F4a2.</td>
<td>Linear Systems: Real Non-Symmetric Banded</td>
</tr>
<tr>
<td>F4a4.</td>
<td>Linear Systems: Real Non-Symmetric Sparse</td>
</tr>
<tr>
<td>F4b1a.</td>
<td>Linear Systems: Real Symmetric Full Indefinite</td>
</tr>
<tr>
<td>F4b1b.</td>
<td>Linear Systems: Real Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td>F4b2.</td>
<td>Linear Systems: Real Symmetric Banded</td>
</tr>
<tr>
<td>F4b4.</td>
<td>Linear Systems: Real Symmetric Sparse</td>
</tr>
<tr>
<td>F4c1.</td>
<td>Linear Systems: Complex Non-Hermitian Full</td>
</tr>
<tr>
<td>F4c2.</td>
<td>Linear Systems: Complex Non-Hermitian Banded</td>
</tr>
<tr>
<td>F4d1a.</td>
<td>Linear Systems: Complex Hermitian Full Indefinite</td>
</tr>
<tr>
<td>F4d1b.</td>
<td>Linear Systems: Complex Hermitian Full Positive-Definite</td>
</tr>
<tr>
<td>F4d2.</td>
<td>Linear Systems: Complex Hermitian Banded</td>
</tr>
<tr>
<td>F4e1.</td>
<td>Linear Systems: Double-Precision Non-Symmetric Full</td>
</tr>
<tr>
<td>F4e2.</td>
<td>Linear Systems: Double-Precision Non-Symmetric Banded</td>
</tr>
<tr>
<td>F4f1a.</td>
<td>Linear Systems: Double-Precision Symmetric Full Indefinite</td>
</tr>
<tr>
<td>F4f1b.</td>
<td>Linear Systems: Double-Precision Symmetric Full Positive-Definite</td>
</tr>
<tr>
<td>F4f2.</td>
<td>Linear Systems: Double-Precision Symmetric Banded</td>
</tr>
<tr>
<td>F5.</td>
<td>QR (Orthogonal) Decomposition</td>
</tr>
<tr>
<td>F6.</td>
<td>Singular Value Decomposition, Rank Determination, Generalized Inverse</td>
</tr>
<tr>
<td>F7.</td>
<td>Updating Matrix Decompositions</td>
</tr>
<tr>
<td>G1.</td>
<td>Fast Fourier Transforms, Sine and Cosine Transforms</td>
</tr>
<tr>
<td>G2.</td>
<td>Convolutions</td>
</tr>
<tr>
<td>G3.</td>
<td>Laplace Transforms</td>
</tr>
<tr>
<td>G4.</td>
<td>Hilbert Transforms</td>
</tr>
<tr>
<td>H1a.</td>
<td>Numerical Differentiation: Function Available only on Grid</td>
</tr>
<tr>
<td>H1b.</td>
<td>Numerical Differentiation: User-Defined Function</td>
</tr>
<tr>
<td>Software Categories</td>
<td>NCAR Software Catalog</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>H2a1.</td>
<td>One-Dimensional Integration: Integrand Defined on a Grid</td>
</tr>
<tr>
<td>H2a2a.</td>
<td>One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand</td>
</tr>
<tr>
<td>H2a2b.</td>
<td>One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand</td>
</tr>
<tr>
<td>H2a3a1.</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, General Integrand</td>
</tr>
<tr>
<td>H2a3a2.</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td>H2a3b1.</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand</td>
</tr>
<tr>
<td>H2a3b2.</td>
<td>One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities</td>
</tr>
<tr>
<td>H2a4.</td>
<td>One-Dimensional Integration: Infinite Interval</td>
</tr>
<tr>
<td>H2b.</td>
<td>Multi-Dimensional Integration</td>
</tr>
<tr>
<td>H2c.</td>
<td>Computation of Weights and Nodes for Quadrature (Integration) Formula</td>
</tr>
<tr>
<td>I1a.</td>
<td>Optimization (Function Minimization): One-Dimensional</td>
</tr>
<tr>
<td>I1b.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Unconstrained</td>
</tr>
<tr>
<td>I2a.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming</td>
</tr>
<tr>
<td>I2h1.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints</td>
</tr>
<tr>
<td>I2h3.</td>
<td>Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints</td>
</tr>
<tr>
<td>J1.</td>
<td>Sequential Input/Output, Including Tape I/O</td>
</tr>
<tr>
<td>J2.</td>
<td>Direct (Random) Access Input/Output</td>
</tr>
<tr>
<td>J3.</td>
<td>Asynchronous (Buffered) Input/Output</td>
</tr>
<tr>
<td>J4.</td>
<td>Free Format and Data Structure Input/Output</td>
</tr>
<tr>
<td>J5a1.</td>
<td>Graphics: Plotting Dashed Lines</td>
</tr>
<tr>
<td>J5a2.</td>
<td>Graphics: Plotting Graphs and Histograms</td>
</tr>
<tr>
<td>J5a3.</td>
<td>Graphics: Representing Two-Dimensional Fields</td>
</tr>
<tr>
<td>J5a4.</td>
<td>Graphics: Representing Objects in Three-Space</td>
</tr>
<tr>
<td>J5a5.</td>
<td>Graphics: Writing Characters</td>
</tr>
<tr>
<td>J5a7.</td>
<td>Graphics: Movie Utilities</td>
</tr>
<tr>
<td>J5b.</td>
<td>Graphics: Other Utilities</td>
</tr>
<tr>
<td>J6.</td>
<td>Miscellaneous I/O, Including Communications, Special Devices</td>
</tr>
<tr>
<td>K1.</td>
<td>File Management: Access, Deletion, and Staging</td>
</tr>
<tr>
<td>K2.</td>
<td>File Copying, Renaming</td>
</tr>
<tr>
<td>K3.</td>
<td>File Editing</td>
</tr>
<tr>
<td>K4.</td>
<td>File Combination, Splitting, and Conversion Tools</td>
</tr>
<tr>
<td>K5.</td>
<td>File Directory, Status, and Size Information</td>
</tr>
<tr>
<td>K6.</td>
<td>File Comparison</td>
</tr>
<tr>
<td>K7.</td>
<td>File Archiving and Version Control</td>
</tr>
<tr>
<td>K8.</td>
<td>File Positioning</td>
</tr>
</tbody>
</table>
K9. File Characteristics and Permissions (Access Control)

L01a. Descriptive Statistics: Location, Dispersion, Shape
L01b. Descriptive Statistics: Distributions, Densities, Frequencies (see also L03a and L03e)

L02. Data Manipulation

L03a. Statistical Graphics: Histograms
L03b. Statistical Graphics: Scatter Plots (see also J5a2)
L03c. Statistical Graphics: Symbol Plots
L03d. Statistical Graphics: Probability Plots
L03e. Statistical Graphics: Exploratory Data Analysis Graphics
L03f. Statistical Graphics: Time Series Plots
L03g. Statistical Graphics: Other Techniques

L04a. Confidence Intervals, Hypothesis Testing
L04b. Nonparametric (Distribution-Free) Analysis (see also L07c, L11a, L11b)
L04c. Goodness-of-Fit Tests (see also L09)

L05a. Statistical Distributions: Beta
L05b. Statistical Distributions: Binomial
L05d. Statistical Distributions: Chi-squared
L05e. Statistical Distributions: Circular Coverage Function
L05i. Statistical Distributions: F Distribution
L05j. Statistical Distributions: Gamma
L05m. Statistical Distributions: Hypergeometric
L05n. Statistical Distributions: Kolmogorov-Smirnov
L05r. Statistical Distributions: Negative Binomial
L05s. Statistical Distributions: Normal
L05u. Statistical Distributions: Poisson
L05v. Statistical Distributions: T Distribution
L05y. Statistical Distributions: General Distribution

L06a. Pseudo-Random Number Generators: Beta
L06b. Pseudo-Random Number Generators: Binomial
L06c. Pseudo-Random Number Generators: Cauchy
L06d. Pseudo-Random Number Generators: Chi-squared
L06g. Pseudo-Random Number Generators: Exponential
L06i. Pseudo-Random Number Generators: F Distribution
L06j. Pseudo-Random Number Generators: Gamma
L06k. Pseudo-Random Number Generators: Geometric
L06m. Pseudo-Random Number Generators: Hypergeometric
L06p. Pseudo-Random Number Generators: Logistic
L06q. Pseudo-Random Number Generators: Lognormal
L06r. Pseudo-Random Number Generators: Negative Binomial
L06s. Pseudo-Random Number Generators: Normal
L06u. Pseudo-Random Number Generators: Poisson
L06v. Pseudo-Random Number Generators: T Distribution
Software Categories

L06w. Pseudo-Random Number Generators: Uniform and Associated Uniform Generator Tests
L06x. Pseudo-Random Number Generators: Weibull
L06y. Pseudo-Random Number Generators: General Distribution
L06z. Pseudo-Random Number Generators: Other Distributions

L07a. Generation of Experimental Designs
L07b. Analysis of Variance
L07c. Nonparametric Analysis of Variance
L07d. Analysis of Covariance
L07e. Multivariate Analysis of Variance
L07f. Tests to use with Analysis of Variance and Covariance

L08a1. Regression: Simple Linear
L08a2. Regression: Polynomial
L08a3. Regression: Multiple Linear
L08a4. Regression: Robust
L08b. Regression: Nonlinear
L08c. Regression: Other
L08d. Techniques Used with Regression

L09. Categorical Data Analysis (Discrete Data Analysis) (see also L01b)

L10a. Time Series: Transformations, Smoothing, Filtering
L10b. Time Series: Autocorrelation and Cross-Correlation, Univariate and Multivariate
L10c. Time Series: ARMA and ARIMA Modeling and Forecasting
L10d. Time Series: Transfer Function Modeling
L10e. Time Series: Univariate Spectral Analysis
L10f. Time Series: Multivariate Spectral Analysis
L10l. Time Series: Other Time Series Techniques (see also L04a)

L11a. Correlation Analysis: No Missing Values
L11b. Correlation Analysis: Missing Values

L12. Discriminant Analysis

L13a. Factor Analysis
L13b. Principal Component Analysis
L13c. Canonical Correlation Analysis

L14. Cluster Analysis

L15. Life Testing, Survival Analysis

L16. Exploratory Data Analysis

L17. Sampling Techniques

L20. Other Statistical Techniques
<table>
<thead>
<tr>
<th>Software Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1a1. Sorting of Integer Data</td>
</tr>
<tr>
<td>M1a2. Sorting of Real Data</td>
</tr>
<tr>
<td>M1a3. Sorting of Character Data</td>
</tr>
<tr>
<td>M2. Conversion, Packing and Unpacking</td>
</tr>
<tr>
<td>M4. Character Manipulation, Bit Operations</td>
</tr>
<tr>
<td>M5. Searching</td>
</tr>
<tr>
<td>N. Debugging</td>
</tr>
<tr>
<td>Q1. Date and Time, Program Timers</td>
</tr>
<tr>
<td>Q3. Machine and System Dependent Constants</td>
</tr>
<tr>
<td>Q4. Documentation Retrieval</td>
</tr>
<tr>
<td>Q5. Error Handling Packages</td>
</tr>
<tr>
<td>Q6. Storage Allocation, Memory Management, and Overlays</td>
</tr>
<tr>
<td>Q7. Job Control, Job Status, Operating System Requests</td>
</tr>
<tr>
<td>S. Software Development Tools, Language Processors</td>
</tr>
<tr>
<td>Z. Non-User-Callable Support Software</td>
</tr>
</tbody>
</table>
A. Arithmetic, Elementary Operations on Polynomials

**IMSL**

**VDCPS** Decompose an integer into its prime factors.

**LOCLIB**

**MP** R. Brent's package for multiple-precision arithmetic. Many special functions are included.

**NAG**

**A02AAF** Evaluates the square root of a complex number.

**A02ABF** Returns the value of the modulus of a complex number.

**A02ACF** Divides one complex number by a second complex number.

**C06BAF** Accelerates the convergence of a given convergent sequence to its limit, using the epsilon algorithm of Wynn.

**C06GBF** Forms the complex conjugate of a Hermitian sequence of $N$ data values.

**C06GCF** Forms the complex conjugate of a sequence of $N$ data values.

**E02RBF** Evaluates a rational function at a user-supplied point, given the numerator and denominator coefficients.

**NCARLB**

**AVPOLY** AVPOLY evaluates a polynomial expression at several ($<65$) values of the independent variable.

**PORT**

**CDADD** Complex double precision addition.

**CDDIV** Complex double precision division.

**CDMUL** Complex double precision multiplication.

**CDSUB** Complex double precision subtraction.

**CEIL** Real function that returns the ceiling of a floating-point number $x$, that is, the smallest integer greater than or equal to $x$.

**DCEIL** Double precision version of CEIL.

**DFLR** Double precision function that returns the floor of a double precision floating-point number $x$, that is, the largest integer less than or equal to $x$. 

Arithmetic, Elementary Operations on Polynomials
DMKFL  Double precision version of MKFL.
DUMKFL Double precision version of UMKFL.
DVBTOV Double precision version of VBTOD.
DVTOD Double precision version of VTOB.
FLR    Real function that returns the floor of a floating-point number $x$, that is, the largest integer less than or equal to $x$.
ICEIL  Integer function that returns the ceiling of a floating-point number $x$, that is, the smallest integer greater than or equal to $x$.
IDCEIL Double precision version of ICEIL.
IDFLR  Integer function that returns the floor of a double precision floating-point number $x$, that is, the largest integer less than or equal to $x$.
IFLR   Integer function that returns the floor of a floating-point number $x$, that is, the largest integer less than or equal to $x$.
MKFL   Make floating-point number. Given $M$, the mantissa of a floating-point number, and $E$, its exponent, MKFL returns $M \cdot B^E$, where $B$ is the base of the machine.
UMKFL  Decompose a floating-point number. Decomposes (“unmakes”) a non-zero floating-point number, $F$, into a mantissa, $M$, and an exponent, $E$, so that $F = M \cdot B^E$, where $B$ is the base of the machine and $\frac{1}{B} \leq M < 1$.
VBTOD  Convert floating-point number to base-10 mantissa and exponent. Converts a machine based representation (mantissa and exponent) of a floating-point number into a normalized base-10 representation.
VDTOB  Convert base-10 number to machine-base. Converts a base-10 representation (mantissa and exponent) of a floating point number into a normalized machine-base representation.

SCILIB

CSUM   Sums the elements of a complex vector. This routine has been optimized for the Cray 1.
FOLRN  Cray 1 optimized routine solving for the last term of the first-order linear recurrence $r_i = b_i$, $r_i = -a_i r_{i-1} + b_i$, $i = 2, 3, \ldots, n$. This routine allows for efficient evaluation of polynomials using Horner’s method.
SSUM   Sums the elements of a real vector. This routine has been optimized for the Cray 1.

SLATEC

BINOM  Computes the binomial coefficients.
CARG   Computes the argument of a complex number.
CBRT   Computes the cube root of $z$.
CCBRT  Computes the complex cube root of $z$.
DBINOM Computes the double precision binomial coefficients.
DCBRT Computes the double precision cube root.
DFAC Computes the double precision factorial of $N$.
FAC Computes the factorial of $N$.

**UTILITIES**

DC VOS Software Tools command for integer ‘desk’ calculation.
B1. Trigonometric and Inverse Trigonometric Functions

NAG

S07AAF Returns the value of the circular tangent, \( \tan(x) \).

S09AAF Returns the value of the inverse circular sine, \( \arcsin(x) \). The value is in the principal range \((-\frac{\pi}{2}, \frac{\pi}{2})\).

S09ABF Returns the value of the inverse circular cosine, \( \arccos(x) \); the result is in the principal range \((0, \pi)\).

PORT

ARCOS Computes the arccosine function.

ARSin Computes the arcsine function.

DARCOS Double precision version of ARCOS.

DARSIN Double precision version of ARSIN.

DTAN Double precision version of TAN.

DTRIGP Double precision version of TRIGP.

TAN Computes the tangent function.

TRIGP Evaluates a trigonometric polynomial given the degree, coefficients, and argument, by a vector variation of Horner’s rule which is numerically stable and efficient.

SLATEC

CACOS Computes the complex arc cosine.

CASIN Computes the complex arc sine.

CATAN Computes the complex arc tangent.

CATAN2 Computes the complex arc tangent in the proper quadrant.

CCOT Computes the complex cotangent.

COSDG Computes the cosine of an argument in degrees.

COT Computes the cotangent.

CTAN Computes the complex tangent.

D9ATN1 Evaluates \( \text{datan}(x) \) from first order relative accuracy so that \( \text{datan}(x) = x + x^3 \cdot D9ATN1(x) \).

DCOSDG Computes the double precision cosine for degree arguments.

DCOT Computes the double precision cotangent.
NCAR Software Catalog

B2. Hyperbolic and Inverse Hyperbolic Functions

**NAG**

- **S10AAF** Returns a value for the hyperbolic tangent, tanh(x).
- **S10ABF** Returns the value of the hyperbolic sine, sinh(x).
- **S10ACF** Returns the value of the hyperbolic cosine, cosh(x).
- **S11AAF** Returns the value of the inverse hyperbolic tangent, arctanh(x).
- **S11ABF** Returns the value of the inverse hyperbolic sine, arcsinh(x).
- **S11ACF** Returns the value of the inverse hyperbolic cosine, arccosh(x). The result is in the principal positive branch.

**PORT**

- **ACOSH** Computes the hyperbolic arccosine function arccosh.
- **ASINH** Computes the hyperbolic arcsine function.
- **ATANH** Computes the hyperbolic arctangent function.
- **COSH** Computes the hyperbolic cosine function.
- **DACOSH** Double precision version of ACOSH.
- **DASINH** Double precision version of ASINH.
- **DATANH** Double precision version of ATANH.
- **DCOSH** Double precision version of COSH.
- **DSINH** Double precision version of SINH.
- **DTANH** Double precision version of TANH.
- **SINH** Computes the hyperbolic sine function.
- **TANH** Computes the hyperbolic tangent function.
SLATEC

ACOSH  Computes the arc hyperbolic cosine.
ASINH  Computes the arc hyperbolic sine.
ATANH  Computes the arc hyperbolic tangent.
CACOSH Computes the complex arc hyperbolic cosine.
CASINH Computes the complex arc hyperbolic sine.
CATANH Computes the complex arc hyperbolic tangent.
CCOSH  Computes the complex hyperbolic cosine.
CSINH  Computes the complex hyperbolic sine.
CTANH  Computes the complex hyperbolic tangent.
DACOSH Computes the double precision arc hyperbolic cosine.
DASINH Computes the double precision arc hyperbolic sine.
DATANH Computes the double precision arc hyperbolic tangent.

B3. Exponential and Logarithmic Functions

PORT

CDEXP  Computes $e^z$ for complex double-precision $z$.
CDLOG Computes $\ln(z)$ for complex double-precision $z$.

SLATEC

ALNREL Evaluates $\ln(1+z)$ accurate in the sense of relative error.
C9LN2R Evaluates $\text{clog}(1+z)$ from second order with relative error so that $\text{clog}(1+z) = z - \frac{z^2}{2} + z^3 \text{C9LN2R}(z)$.
CEXPRL Evaluates $\frac{e^z-1}{z}$ so that $e^z = 1+z \text{ CEXPRL}(z)$.
CLNREL Computes the principal value of the complex natural logarithm of $1+z$ with relative error accuracy for small $|z|$.
CLOG10 Computes the principal value of the complex base 10 logarithm.
D9LN2R Evaluates $\text{dlog}(1+z)$ from second order relative accuracy so that $\text{dlog}(1+z) = z - \frac{z^2}{2} + z^3 \text{D9LN2R}(z)$.
DEXPRL Calculates the double precision relative error exponential $\frac{e^z-1}{z}$.
DLNREL Evaluates double precision $\ln(1+x)$ accurate in the relative error sense.

EXPREL Evaluates $\text{EXPREL}(x) = \frac{e^x - 1}{x}$.

R9LN2R Evaluates $\log(1+x)$ from second order relative accuracy so that
$\log(1+x) = x - \frac{x^2}{2} + x^3 \cdot \text{R9LN2R}(x)$.

B5e. Exponential, Logarithmic, Sine and Cosine Integrals

**AMOSLIB**

**COSINT** Computes $\int_{x}^{\infty} \frac{\cos(t)}{t} \, dt$.

**SININT** Computes $\int_{0}^{x} \frac{\sin(t)}{t} \, dt$ and $\int_{x}^{\infty} \frac{\sin(t)}{t} \, dt$.

**FUNPACK**

**EI** Computes single precision values of the exponential integral $Ei(x) = \int_{-\infty}^{x} \frac{e^t}{t} \, dt$ for $x > 0$ and $Ei(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t} \, dt$ for $x < 0$.

**EONE** Computes single precision values of the exponential integrals $E_1(x) = -Ei(-x)$ where $Ei(x) = \int_{-\infty}^{x} \frac{e^t}{t} \, dt$ for $x > 0$ and $Ei(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t} \, dt$ for $x < 0$.

**EXPEI** Computes single precision values of the exponential integrals $e^{-x}Ei(x)$ where $Ei(x) = \int_{-\infty}^{x} \frac{e^t}{t} \, dt$ for $x > 0$ and $Ei(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t} \, dt$ for $x < 0$.

**IMSL**

**MMDEI** Exponential integrals.

**MMDEN** Exponential integrals of integer order for real argument $x$ scaled by $e^x$.

**NAG**

**S13AAF** Returns the value of the exponential integral.

**S13ACF** Returns the value of the cosine integral.

**S13ADF** Returns the value of the sine integral.

**S15AFF** Returns a value for Dawson's Integral, $F(x)$.

**S20ACF** Returns a value for the Fresnel Integral, $S(x)$.

**S20ADF** Returns a value for the Fresnel Integral, $C(x)$.

47 Elementary and Special Functions
**SLATEC**

**ALI** Computes the logarithmic integral.

**DE1** Computes the double precision exponential integral $E_1(x)$.

**DEI** Computes the double precision exponential integral $E_i(x)$.

**DEXINT** Computes $m$ member sequences of exponential integrals $E_{n+k}(x)$, $k = 0, 1, \ldots, m-1$ for $n \geq 1$ and $x \geq 0$. Double precision version.

**DLI** Computes the double precision logarithmic integral.

**DSPENC** Computes a double precision form of Spence's integral due to K. Mitchell.

**EI** Computes the exponential integral $E_1(x)$.

**EI** Computes the exponential integral $E_i(x)$.

**EXINT** Computes $m$ member sequences of exponential integrals $E_{n+k}(x)$, $k = 0, 1, \ldots, m-1$ for $n \geq 1$ and $x \geq 0$.

**SPENC** Computes Spence's function.

---

**B5f. Gamma, Beta, Psi Functions**

**AMOSLIB**

**BETAIC** Computes an $n$ member sequence of beta distributions $y(k) = I_x(a + k - 1, b)$, $k = 1, \ldots, n$, $a > 0$, $b > 0$, and $0 \leq x \leq 1$, where $I_x(a, b)$ is the incomplete beta function normalized to 1.

**BETALN** Computes the natural log of the complete beta function, $\ln B(a, b)$, where $B(a, b)$ is defined in terms of the gamma function by $B(a, b) = \frac{\Gamma(a) \Gamma(b)}{\Gamma(a + b)}$.

**GAMFN** Computes the gamma function on the interval (-169,+170) except at zero or the negative integers, where the gamma function has poles.

**GAMIC** Computes an $n$ member sequence of incomplete gamma functions normalized so that at $x = \infty$, the incomplete gamma function has the value 1.

**GAMLN** Computes the natural log of the gamma function for $x > 0$.

**GAMTL** Evaluates an $n$ member sequence of complementary gamma functions $y(k) = \frac{\gamma(b + k - 1, x)}{\Gamma(b + k - 1)}, k = 1, \ldots, n$, to a specified relative error for $x \geq 0$ and $b > 0$. 

---

Elementary and Special Functions 48
**FUNPACK**

**PSI**
Computes single precision values of the logarithmic derivative of the gamma function $ψ(x) = \frac{d\Gamma(x)}{dz} = \frac{d(\ln \Gamma(x))}{dz}$, for real arguments $x$ where $|x| \geq 2^{-275}$, $-x < 2^9$, and $x$ not a negative integer.

**IMSL**

**ALGAMA**
(=MLGAMA) Evaluate the log (base $e$) of the absolute value of the gamma function.

**GAMMA**
(=MGAMA) Evaluate the gamma function.

**MMPSI**
Logarithmic derivative of the gamma function.

**NAG**

**S14AAF**
Returns the value of the Gamma function.

**S14ABF**
Returns a value for the logarithm of the Gamma function, $\ln(\Gamma(x))$.

**SAS (IBM 4341)**

**DIGAMMA**

**GAMMA**

**LGAMMA**

**SLATEC**

**ALBETA**
Computes the natural log of the complete beta function.

**ALGAMS**
Computes $\log |\Gamma(x)|$.

**ALNGAM**
Computes the log of the absolute value of the Gamma function.

**BETA**
Computes the complete beta function.

**BETAI**
Computes the incomplete beta function.

**C0LGMC**
Evaluates $(z + \frac{1}{2})\log (\frac{z+1}{z})-1$ with relative accuracy.

**C9LGMC**
Computes the log Gamma correction term for most $z$ so that $\log (\text{egamma}(z)) = \frac{1}{2} \log (2\pi) + (z-\frac{1}{2})\log (z) - z + C9LGMC(z)$.

**CBETA**
Computes the complete beta function of complex parameters $A$ and $B$.

**CGAMMA**
Computes the gamma function of complex argument.

**CGAMR**
Computes the reciprocal gamma function of complex argument.

**CLBETA**
Computes the natural log of the complex valued complete beta function of complex parameters $A$ and $B$.
CLNGAM  Computes the natural log of the complex valued Gamma function of complex argument.

CPSI  Computes the psi function of complex argument.

D9GMIC  Calculates the double precision incomplete gamma function for a near a negative integer and z small.

D9GMIT  Computes double precision Tricomi's incomplete gamma function for small z.

D9LGIC  Computes the double precision log incomplete gamma function for large x and for a ≤ z.

D9LGIT  Computes the log of Tricomi's incomplete gamma function with Perron's continued fraction for large x and a ≥ z.

D9LGMC  Computes the double precision log Gamma correction factor for x > 10, so that dlog(dgamma(x)) = dlog(dsqrt(2π)) + (x - 1/2)dlog(x) - x + D9LGMC(x).

DBETA  Computes the double precision complete beta function.

DBETAI  Calculates the double precision incomplete beta function.

DGAMI  Calculates the double precision incomplete gamma function.

DGAMIC  Calculates the double precision complementary incomplete gamma function.

DGAMIT  Calculates Tricomi's form of the incomplete gamma function. Double precision version.

DGAMLMLM  Computes the double precision minimum and maximum bounds for x in GAMMA(z).

DGAMMA  Computes the double precision complete gamma function.

DGAMR  Calculates double precision reciprocal gamma function.

DLBETA  Computes the double precision natural logarithm of the complete Beta function.

DLGAMS  Calculates the log of the absolute value of the Gamma function. Double precision version.

DLNGAM  Computes the double precision logarithm of the absolute value of the gamma function.

DPSI  Computes the double precision Psi (or Digamma) function.

GAMI  Computes the incomplete gamma function.

GAMIC  Computes the complementary incomplete gamma function.

GAMIT  Computes Tricomi's form of the incomplete gamma function.

GAMLIM  Computes the minimum and maximum bounds for x in GAMMA(z).

GAMMA  Computes the gamma function.

GAMR  Computes the reciprocal of the gamma function.

PSI  Computes the Psi (or Digamma) function.

R9GMIC  Computes the complementary incomplete gamma function for a near a negative integer and for small x.

R9GMIT  Computes Tricomi's incomplete gamma function for small x.
**R9LGIC** Computes the log complementary incomplete gamma function for large $x$ and for $a \leq x$.

**R9LGIT** Computes the log of Tricomi's incomplete gamma function with Perron's continued fraction for large $x$ and $a \geq x$.

**R9LGMC** Computes the log Gamma correction factor so that $\log(\Gamma(x)) = \log(\sqrt{2\pi}) + (x - \frac{1}{2}) \log(x) - x + R9LGMC(x)$.

---

**B5g.  Error Functions, Voigt Functions**

**AMOSLIB**

**CERF** Computes the error function $\text{erf}(z)$, or the scaled coerror function $e^{-z^2}\text{erfc}(z)$, for complex $z$.

**ERF** Computes the error function $\frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \, dt$. The approximation used is correct to 11 significant digits.

**ERFC** Computes the complementary error function $\frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-t^2} \, dt$ correct to 11 significant digits.

**INERFC** Computes $n$-iterated coerror functions (repeated integrals of the coerror function $\text{erfc}(x)$) $y(k) = i^k \text{erfc}(x)$, $k = n_0, n_0+1, \ldots, n_0+n-1$ or scaled functions $y(k) = i^k \text{erfc}(x) e^{-z^2}$, $k = n_0, n_0+1, \ldots, n_0+n-1$ for $x > 0$.

**RERF** Computes the error function and the coerror function for real $x$.

**RIERFC** Computes the inverse coerror function.

**FUNPACK**

**DAW** Computes single precision values of Dawson's integral $e^{-z^2} \int_0^z e^{t^2} \, dt$, defined for all real arguments.

**IMSL**

**ERF** (=MERF) Evaluate the error function.

**ERFC** (=MERRC) Evaluate the complemented error function.

**MERFCI** Inverse complemented error function.

**MERFI** Inverse error function.

**MERRCZ** Evaluate a function related to the complemented error function for a complex argument.

**MMDAS** Dawson integral.
MSMRAT  Ratio of the ordinate to the upper tail area of the standardized normal (Gaussian) distribution.

**LOCLIB**

VOIGTD  Computes the Voigt function accurate to at least 4 significant digits for a wide range of arguments. Runs about 5 times faster than VOIGTR.

VOIGTR  Computes the Voigt function accurate to 8 significant digits over a wide range of arguments.

**NAG**

S15ADF  Returns the value of the complementary error function, erfc(x).

S15AEF  Returns the value of the error function, erf(x).

**NCARLB**

ERFS  Calculates the error function and complementary error function of a real argument.

**SAS (IBM 4341)**

ERF  Function which computes the error function. Can be used to find the probability that a normally distributed random variable with mean 0 and standard deviation 1 will take on a values less than X. Documented in *SAS User's Guide: Basics - Version 5*.

ERFC  Function which computes the complement to the ERF function (1-ERF). Documented in *SAS User's Guide: Basics - Version 5*.

**SLATEC**

DERF  Computes the double precision error function, ERF, of x.

DERFC  Computes the double precision complementary error function, ERFC.

ERF  Computes the error function, ERF.

ERFC  Computes the complementary error function (ERFC).
B51. I Bessel Functions of Real Argument

**AMOSLIB**

**BESI** Computes an $n$ member sequence of $I$ Bessel functions $I_{\alpha+k-1}(x), k = 1, \ldots, n$ or scaled Bessel functions $e^{-x}I_{\alpha+k-1}(x), k = 1, \ldots, n$ for non-negative $\alpha$ and $x$.

**BESIO1** Computes Bessel functions $I_{\nu}(x)$, $\nu = 0$ or 1 or scaled Bessel functions $e^{-|x|}I_{\nu}(x)$, $\nu = 0$ or 1 for real $x$.

**SBESI** Computes a sequence of scaled $I$ Bessel functions for real, non-negative arguments. Except for scaling, the manner of computation is the same as that for BESI.

**FUNPACK**

**BESEIO** Computes approximate values for the modified Bessel function of the first kind of order zero multiplied by $e^{|x|}$, where $x$ is any single precision argument.

**BESEI1** Computes approximate values for the modified Bessel function of the first kind of order one multiplied by $e^{|x|}$, where $x$ is any single precision argument.

**BESIO** Computes approximate values for the modified Bessel function of the first kind of order zero for arguments less than or equal to 745.894. Single precision.

**BESI1** Computes approximate values for the modified Bessel function of the first kind of order one for arguments with absolute value less than or equal to 745.894.

**IMSL**

**MMBSSI0** Modified Bessel function of the first kind of order zero.

**MMBSSI1** Modified Bessel function of the first kind of order one.

**MMBSIN** Modified Bessel function of the first kind of nonnegative integer order for real arguments.

**MMBSIR** Modified Bessel function of the first kind of nonnegative real order for real positive arguments with exponential scaling option.

**LOCLIB**

**BESLRI** Double precision version of BESR on LOCLIB.

**NAG**

**S18AEF** Returns the value of the modified Bessel Function, $I_0(x)$.

**S18AFF** Returns the value of the modified Bessel Function, $I_1(x)$.

**S18CEF** Computes the value of the scaled modified Bessel function $e^{-|x|}I_0(x)$ for real $x$.

**S18CFF** Computes the value of the scaled modified Bessel function $e^{-|x|}I_1(x)$ for real $x$. 
**NCARLB**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BESR</td>
<td>Calculates Bessel functions $I$ and $J$ for real argument and integer orders.</td>
</tr>
<tr>
<td>CODY</td>
<td>Calculates Bessel functions $I_{n+\alpha}(x)$ for non-negative argument $x$, and non-negative order $n+\alpha$ with or without exponential scaling.</td>
</tr>
</tbody>
</table>

**PORT**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BESRI</td>
<td>Calculates the modified Bessel functions, $I_k(z)$, for $z$ real, and $k=0,1,\ldots,N$.</td>
</tr>
<tr>
<td>DBESRI</td>
<td>Double precision version of BESRI.</td>
</tr>
</tbody>
</table>

**SLATEC**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BESI</td>
<td>Computes an $n$ member sequence of $I$ Bessel functions $I_{\alpha+k-1}(x)$, $k=1,\ldots,n$ or scaled Bessel functions $e^{-x}I_{\alpha+k-1}(x)$, $k=1,\ldots,n$ for non-negative $\alpha$ and $x$.</td>
</tr>
<tr>
<td>BESIO</td>
<td>Computes the hyperbolic Bessel function of the first kind of order zero.</td>
</tr>
<tr>
<td>BESIOE</td>
<td>Computes the exponentially scaled hyperbolic Bessel function of the first kind of order zero.</td>
</tr>
<tr>
<td>BESI1</td>
<td>Computes the hyperbolic Bessel function of first kind of order one.</td>
</tr>
<tr>
<td>BESI1E</td>
<td>Computes the exponentially scaled hyperbolic Bessel function of the first kind of order one.</td>
</tr>
<tr>
<td>DBESI</td>
<td>Computes an $n$ member sequence of $I$ Bessel functions $I_{\alpha+k-1}(x)$, $k=1,\ldots,n$ or scaled Bessel functions $e^{-x}I_{\alpha+k-1}(x)$, $k=1,\ldots,n$ for nonnegative $\alpha$ and $x$. Double precision version.</td>
</tr>
<tr>
<td>DBESIO</td>
<td>Computes the double precision hyperbolic Bessel function of the first kind of order zero.</td>
</tr>
<tr>
<td>DBESI1</td>
<td>Computes the double precision modified (hyperbolic) Bessel function of the first kind of order one.</td>
</tr>
<tr>
<td>DBESI1E</td>
<td>Computes the double precision exponentially scaled hyperbolic Bessel function of the first kind of order zero.</td>
</tr>
<tr>
<td>DBSIOE</td>
<td>Computes the double precision exponentially scaled modified (hyperbolic) Bessel function of the first kind of order one.</td>
</tr>
</tbody>
</table>
B5i2.  K Bessel Functions of Real Argument

**AMOSLIB**

**BESKO1** Computes Bessel functions \( K_v(x), \nu=0 \) or 1 or scaled Bessel functions \( e^x K_v(x), \nu=0 \) or 1 for \( x > 0 \).

**BESKN** Computes a sequence of integer order Bessel functions \( K_{\nu+i-1}(x), i=1, \ldots, n \) for real \( x > 0 \) and a non-negative integer \( \nu \).

**FUNPACK**

**BESEK0** Computes approximate values for the modified Bessel function of the second kind of order zero multiplied by the exponential function, for arguments less than or equal to \( 2^{-975} \).

**BESEK1** Computes approximate values for the modified Bessel function of the second kind of order one multiplied by the exponential function, for arguments less than \( 2^{-975} \).

**BESK0** Computes approximate values for the modified Bessel function of the second kind of order zero for arguments in the inclusive range \( 2^{-975} \) through 672.788.

**BESK1** Computes approximate values for the modified Bessel function of the second kind of order one for arguments in the inclusive range \( 2^{-975} \) through 672.789.

**IMSL**

**MMBSK0** Modified Bessel function of the second kind of order zero.

**MMBSK1** Modified Bessel function of the second kind of order one.

**MMBSKR** Modified Bessel function of the second kind of nonnegative real fractional order for real positive arguments scaled by \( e^x \).

**NAG**

**S18ACF** Returns the value of the modified Bessel Function, \( K_0(x) \).

**S18ADF** Returns the value of the modified Bessel Function, \( K_1(x) \).

**S18CCF** Computes the value of the scaled modified Bessel function \( e^x K_0(x) \) for real \( x \).

**S18CDF** Computes the value of the scaled modified Bessel function \( e^x K_1(x) \) for real \( x \).

**SLATEC**

**BESK** Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions \( K_{\nu+i-1}(x) \), or scaled Bessel functions \( e^x K_{\nu+i-1}(x), i=1, \ldots, n \) for real \( x > 0 \) and non-negative orders \( \nu \).

**BESK0** Computes the hyperbolic Bessel function of the third kind of order zero.

**BESK0E** Computes the exponentially scaled hyperbolic Bessel function of the third kind of order zero.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BESK1</strong></td>
<td>Computes the hyperbolic Bessel function of the third kind of order one.</td>
</tr>
<tr>
<td><strong>BESK1E</strong></td>
<td>Computes the exponentially scaled hyperbolic Bessel function of the third kind of order one.</td>
</tr>
<tr>
<td><strong>BESKES</strong></td>
<td>Computes a sequence of exponentially scaled modified Bessel functions of the third kind of fractional order.</td>
</tr>
<tr>
<td><strong>BESKS</strong></td>
<td>Computes a sequence of modified Bessel functions of the third kind of fractional order.</td>
</tr>
<tr>
<td><strong>D0KNUS</strong></td>
<td>Computes Bessel functions $e^z K_\nu(x)$ and $e^z K_{\nu+1}(x)$ for $0 \leq \nu &lt; 1$ Double precision version.</td>
</tr>
<tr>
<td><strong>DBESK</strong></td>
<td>Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $K_{\nu+i-1}(x)$, or scaled Bessel functions $e^z K_{\nu+i-1}(x)$, $i=1, \ldots, n$ for real $x &gt; 0$ and non-negative orders $\nu$. Double precision version.</td>
</tr>
<tr>
<td><strong>DBESK0</strong></td>
<td>Computes double precision modified (hyperbolic) Bessel function of the third kind of order zero.</td>
</tr>
<tr>
<td><strong>DBESK1</strong></td>
<td>Computes the double precision modified Bessel function of the third kind of order one.</td>
</tr>
<tr>
<td><strong>DBESKS</strong></td>
<td>Computes a double precision sequence of modified Bessel functions of the third kind of fractional order.</td>
</tr>
<tr>
<td><strong>DBSK0E</strong></td>
<td>Computes the double precision exponentially scaled modified (hyperbolic) Bessel function of the third kind of order zero.</td>
</tr>
<tr>
<td><strong>DBSK1E</strong></td>
<td>Computes the exponentially scaled, modified (hyperbolic) Bessel function of the third kind of order one. Double precision version.</td>
</tr>
<tr>
<td><strong>DBSKES</strong></td>
<td>Computes a double precision sequence of exponentially scaled modified Bessel functions of the third kind of fractional order.</td>
</tr>
<tr>
<td><strong>R0KNUS</strong></td>
<td>Computes Bessel functions $e^z K_\nu(x)$ and $e^z K_{\nu+1}(x)$ for $0 \leq \nu &lt; 1$.</td>
</tr>
</tbody>
</table>

**B5j1. J Bessel Functions of Real Argument**

**AMOSLIB**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BESJ</strong></td>
<td>Computes an $n$ member sequence of $J$ Bessel functions $J_{\alpha+k-1}(x)$, $k=1, \ldots, n$ for non-negative $\alpha$ and $x$.</td>
</tr>
<tr>
<td><strong>BESJ01</strong></td>
<td>Computes Bessel functions $J_{\nu}(x)$, $\nu=0$ or $1$ for real unrestricted $x$.</td>
</tr>
<tr>
<td><strong>BESY01</strong></td>
<td>Computes Bessel functions $J_{\nu}(x)$ and $Y_{\nu}(x)$, $\nu=0$ or $1$ for $x &gt; 0$.</td>
</tr>
<tr>
<td><strong>J01BES</strong></td>
<td>Obsolete version of BESJ01 on AMOSLIB.</td>
</tr>
<tr>
<td><strong>Y01BES</strong></td>
<td>Obsolete version of BESY01 on AMOSLIB.</td>
</tr>
</tbody>
</table>
**FUNPACK**

**BESJ0**
Computes single precision values of the Bessel function of the first kind and order zero $J_0(x)$ for real arguments $x$ where $|x|$ is less than or equal to $2^{27}$.

**BESJ1**
Computes single precision values of the Bessel function of the first kind and order one $J_1(x)$ for real arguments $x$ where $|x|$ is less than or equal to $2^{27}$.

**IMSL**

**MMBSJ0**
Bessel function of the first kind of order zero.

**MMBSJ1**
Bessel function of the first kind of order one.

**MMBSJN**
Bessel function of the first kind of nonnegative integer order for real arguments.

**MMBSJR**
Bessel function of the first kind of nonnegative real order for real positive arguments.

**LOCLIB**

**BESLRI**
Double precision version of BESR on LOCLIB.

**NAG**

**S17AEF**
Returns the value of the Bessel Function, $J_0(x)$.

**S17AFF**
Returns the value of the Bessel Function, $J_1(x)$.

**NCARLB**

**BESR**
Calculates Bessel functions $I$ and $J$ for real argument and integer orders.

**PORT**

**BESRJ**
Calculates the Bessel functions, $J_k(x)$, for $x$ real, and $k = 0, 1, \ldots, N$.

**DBESRJ**
Double precision version of BESRJ.

**SLATEC**

**BESJ**
Computes an $n$ member sequence of $J$ Bessel functions $J_{\alpha+k-1}(x)$, $k = 1, \ldots, n$ for non-negative $\alpha$ and $x$.

**BESJ0**
Computes the Bessel function of the first kind of order zero.

**BESJ1**
Computes the Bessel function of the first kind of order one.

**D9B0MP**
Evaluates the double precision modulus and phase for $J_0$ and $Y_0$ Bessel functions.

**D9B1MP**
Evaluates the double precision modulus and phase for the $J_1$ and $Y_1$ Bessel functions.

**DBESJ**
Computes an $n$ member sequence of $J$ Bessel functions $J_{\alpha+k-1}(x)$, $k = 1, \ldots, n$ for non-negative $\alpha$ and $x$. Double precision version.

**DBESJ0**
Computes the double precision Bessel function of the first kind of order zero.

Elementary and Special Functions
DBESJ1 Computes the double precision Bessel function of the first kind of order one.

B5j2. Y Bessel Functions of Real Argument

AMOSLIB

BESY01 Computes Bessel functions $J_\nu(x)$ and $Y_\nu(x)$, $\nu=0$ or 1 for $x > 0$.
BESYN Computes a sequence $Y_{\nu+k-1}(x), k = 1, \ldots, n$ of integer order Bessel functions for real $x > 0$ and a non-negative integer $\nu$.
Y01BES Obsolete version of BESY01 on AMOSLIB.

FUNPACK

BESY Computes a sequence of single precision values of the Bessel function of the second kind and non-negative real order $Y_\nu(x)$.

YNU Computes approximate values for the Bessel function of the second kind for real positive argument $x$ in the inclusive range $2^{-128}$ through $2^{27}$, and real fractional order $\nu$, where $\nu$ is greater than or equal to 0.0 and less than or equal to 1.0.

IMSL

MMBSYN Bessel function of the second kind of nonnegative real fractional order for real positive arguments.

NAG

S17ACF Returns the value of the Bessel Function, $Y_0(x)$.
S17ADF Returns the value of the Bessel Function, $Y_1(x)$.

SLATEC

BESY Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y_{\nu+i-1}(x), i = 1, \ldots, n$ for real $x > 0$ and non-negative orders $\nu$.
BESY0 Computes the Bessel function of the second kind of order zero.
BESY1 Computes the Bessel function of the second kind of order one.
D9B0MP Evaluates the double precision modulus and phase for $J_0$ and $Y_0$ Bessel functions.
D9B1MP Evaluates the double precision modulus and phase for the $J_1$ and $Y_1$ Bessel functions.
DBESY Implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y_{\nu+i-1}(x), i = 1, \ldots, n$ for real $x > 0$ and non-negative orders $\nu$. Double precision version.
Computes the double precision Bessel function of the second kind of order zero.
Computes the double precision Bessel function of the second kind of order one.

**B5k. Bessel Functions of Complex Argument, Kelvin Functions**

**AMOSLIB**

**BERBEI** Computes sequences of $\text{ber}_\alpha(x)$ and $\text{bei}_\alpha(x)$ Bessel functions or scaled Bessel functions $e^{\pm z} \text{ber}(z)$ and $e^{\pm z} \text{bei}(z)$ for orders $\alpha, \ldots, \alpha+n-1$ for $x \geq 0$.

**CJYHBS** Computes Bessel functions $J_\nu(z)$, $Y_\nu(z)$, and Struve functions $H_\nu(z)$, for complex $z$ and $\nu=0$ or 1.

**CJO1BS** Computes Bessel functions $J_0(z)$ and $J_1(z)$ of order zero and one respectively for complex $z$.

**IMSL**

**MMBZIN** Modified Bessel function of the first kind of nonnegative integer order for complex arguments.

**MMBZJN** Bessel function of the first kind of nonnegative integer order for complex arguments.

**MMKEL0** Kelvin functions of the first kind, $(\text{ber}, \text{bei})$, and of the second kind, $(\text{ker}, \text{kei})$, of order zero.

**MMKEL1** Kelvin functions of the first kind, $(\text{ber}, \text{bei})$, and of the second kind, $(\text{ker}, \text{kei})$, of order one.

**MMKELD** Derivatives of the Kelvin functions $(\text{ber}, \text{bei}, \text{ker}, \text{kei})$ of order zero.

**LOCLIB**

**BESLCI** Double precision version of BESC on LOCLIB.

**BESSKA** Computes modified Bessel function $K_a(z)$, where $a$ is real and $z$ is complex.

**NAG**

**S19AAF** Returns a value for the Kelvin function $\text{ber } x$.

**S19ABF** Returns a value for the Kelvin function $\text{bei } x$.

**S19ACF** Returns a value for the Kelvin function $\text{ker } x$.

**S19ADF** Returns a value for the Kelvin function $\text{kei } x$. 
NCARLB

<table>
<thead>
<tr>
<th><strong>BESC</strong></th>
<th>Calculates Bessel functions $I$ and $J$ for complex argument and integer orders.</th>
</tr>
</thead>
</table>

PORT

<table>
<thead>
<tr>
<th><strong>BESCI</strong></th>
<th>Calculates the modified Bessel functions, $I_k(z)$, for $z$ complex and $k=0,1,\ldots,N$.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BESCJ</strong></td>
<td>Calculates the Bessel functions, $J_k(z)$, for $z$ complex, and $k=0,1,\ldots,N$.</td>
</tr>
<tr>
<td><strong>DBESCI</strong></td>
<td>Double precision version of BESCI.</td>
</tr>
<tr>
<td><strong>DBESCJ</strong></td>
<td>Double precision of BESCJ.</td>
</tr>
</tbody>
</table>

B51. Miscellaneous Bessel and Related Functions, Airy Functions

AMOSLIB

<table>
<thead>
<tr>
<th><strong>AIRY</strong></th>
<th>Computes the Airy function $Ai(x)$, $x$ real, with an option for scaled values for $x \geq 0$.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BAIRY</strong></td>
<td>Computes the Airy function $Bi(x)$, $x$ real, with an option for scaled values for $x \geq 0$.</td>
</tr>
<tr>
<td><strong>DAIRY</strong></td>
<td>Computes the derivative of the Airy function $Ai(x)$, $x$ real, with an option for scaled values for $x \geq 0$.</td>
</tr>
<tr>
<td><strong>DDBAIRY</strong></td>
<td>Computes the derivative of the Airy function $Bi(x)$, $x$ real, with an option for scaled values for $x \geq 0$.</td>
</tr>
<tr>
<td><strong>DKRKi</strong></td>
<td>Computes the real and imaginary parts of the derivative of $K_0(xe^{i\theta})=dker(x)+i\text{dkei}(x)$ or scaled values $e^{\sqrt{\nu^2-dker(x)}}e^{\sqrt{\nu^2}\text{dkei}(x)}$ for real $x &gt; 0$.</td>
</tr>
<tr>
<td><strong>FINTK0</strong></td>
<td>Computes the integral of the Bessel function $K_0(x)$ on $(0,x)$ or on $(x,\infty)$.</td>
</tr>
<tr>
<td><strong>JROOTS</strong></td>
<td>Computes positive zeros of the $J$ Bessel function of order $\nu$.</td>
</tr>
<tr>
<td><strong>KRKi</strong></td>
<td>Computes the real and imaginary parts of $K_0(xe^{i\theta})=fker(x)+ifkei(x)$ or scaled values $e^{\sqrt{\nu^2}fker(x)},e^{\sqrt{\nu^2}ifkei(x)}$ for real $x &gt; 0$.</td>
</tr>
<tr>
<td><strong>RNU</strong></td>
<td>Computes a ratio of Bessel functions of real, non-negative arguments.</td>
</tr>
</tbody>
</table>

LOCLIB

| **CAIRY** | Computes the complex Airy function and its derivative for arbitrary complex argument; accuracy should be at least seven digits. |

Elementary and Special Functions 60
NCAR Software Catalog

NAG

S17AGF Returns a value for the Airy function, $Ai(x)$.
S17AHF Returns a value of the Airy function, $Bi(x)$.
S17AJF Returns a value of the derivative of the Airy function $Ai(x)$.
S17AKF Returns a value for the derivative of the Airy function $Bi(x)$.

SLATEC

AI Computes the Airy function.
AIE Computes the exponentially scaled Airy function.
BI Computes the Bairy function.
BIE Computes the exponentially scaled Bairy function.
D9AIMP Evaluates the Airy modulus and phase for $x < -1$.
DAI Calculates the double precision Airy function.
DAIE Calculates the double precision Airy function for negative $x$ and an exponentially scaled Airy function for positive $x$.
DBI Calculates the double precision Bairy function (Airy of second kind).
DBIE Calculates the double precision Bairy function for negative $x$ and an exponentially scaled Bairy function for positive $x$.
R9AIMP Evaluates the Airy modulus and phase.

B5m. Elliptic Integrals, Elliptic Functions, Theta Functions

FUNPACK

ELIE1 Computes single precision values of the complete elliptic integrals of the second kind $E(k^2)$ for the parameter $k$ where $E(k^2) = \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2 \phi}} d\phi, 0 \leq k^2 \leq 1$
ELIEM Computes single precision values of the complete elliptic integrals of the second kind $E(1-\eta)$ for the parameter $\eta$ where $E(k^2) = \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2 \phi}} d\phi, 0 \leq \eta \leq 1$
ELIK1 Computes single precision values of the complete elliptic integrals of the first kind $K(k^2)$ for the parameter $k$ where $K(k^2) = \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2 \phi}} d\phi, 0 \leq k^2 \leq 1$
ELIKM Computes single precision values of the complete elliptic integrals of the first kind $K(1-\eta)$ for the parameter $\eta$ where $K(k^2) = \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2 \phi}} d\phi, 0 \leq \eta \leq 1$
ELIPE Computes single precision values of the complete elliptic integrals of the second kind $E(k^2)$ for the parameter $k^2$ where $E(k^2) = \int_0^{\pi/2} \sqrt{1-k^2\sin^2 \phi} d\phi, 0 \leq k^2 \leq 1$
ELIPK Computes single precision values of the complete elliptic integrals of the first kind $K(k^2)$ for the parameter $k^2$ where $K(k^2) = \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2\phi}} \, d\phi$, $0 \leq k^2 \leq 1$

**IMSL**

MMDELE Complete elliptic integral of the second kind.

MMDELK Complete elliptic integral of the first kind.

MMLIND Incomplete elliptic integral of the second kind.

MMLINF Incomplete elliptic integral of the first kind.

MMLINJ Incomplete elliptic integral of the third kind.

**NAG**

S21BAF Returns a value of an elementary integral, which occurs as a degenerate case of an elliptic integral of the first kind.

S21BBF Returns a value of the symmetrised elliptic integral of the first kind.

S21BCF Returns a value of the symmetrised elliptic integral of the second kind.

S21BDF Returns a value of the symmetrised elliptic integral of the third kind.

**B5r. Orthogonal Polynomials, Including Legendre Polynomials**

**IMSL**

RLPOL Generate orthogonal polynomials with the associated constants aa and bb.

**NAG**

C06DBF Returns the value of the sum of a Chebyshev series.

**NCARLB**

ALFPACK Computes the normalized associated Legendre functions PBAR(N,M,THETA) where THETA is colatitude.

OLDALFPK Obsolete package, formerly called ALFPAC. Use ALFPACK instead.

**PORT**

DORTHP Double precision version of ORTHP.

DTCHBP Double precision version of TCHBP.

ORTHPI Evaluates a polynomial expressed as a sum of general orthogonal polynomials using the generalized Clenshaw recurrence scheme.
TCHBP Evaluates a polynomial expressed as a sum of Chebyshev polynomials by Clenshaw's recurrence scheme, which requires only \( n + 1 \) multiplications and \( 2n \) additions for a polynomial of degree \( n \). While the evaluation of this representation requires slightly more work than the evaluation of the power representation by Horner's rule, the Chebyshev polynomial approach is preferable from the point of view of numerical stability.

\[ S L A T E C \]

CSEVL Evaluates the \( N \)-term Chebyshev series at \( x \).

DCSEVL Evaluates the double precision \( N \)-term Chebyshev series \( A \) at \( x \).

INITDS Initializes the double precision properly normalized orthogonal polynomial series to determine the number of terms needed for specific accuracy.

INITS Initializes an orthogonal series so that it defines the number of terms to carry in the series to meet a specified error.

\[ B 5 z . \]

Miscellaneous Higher Mathematical Functions

\[ A M O S L I B \]

HYPGEO Sums the series for the Gauss hypergeometric function \( F(a, b; c; x) \).

THA Computes the \( t(h, a) \) integral of Owen. \( t(h, a) = \int_0^1 e^{-h(1+z^2)/2} \frac{dz}{1+z^2} \).

\[ I M S L \]

MMLINC Elementary integral from which inverse circular functions, logarithms, or inverse hyperbolic functions may be computed.

MMWPL Weierstrass \( p \)-function in the lemniscatic case for complex argument with unit period parallelogram.

MMWPL1 First derivative of the Weierstrass \( P \)-function in the lemniscatic case for complex argument with unit period parallelogram.

MMWPQ Weierstrass \( P \)-function in the equianharmonic case for complex argument with unit period parallelogram.

MMWPQ1 First derivative of the Weierstrass \( P \)-function in the equianharmonic case for complex argument with unit period parallelogram.
SLATEC

CHU Computes the logarithmic confluent hypergeometric function.
D9CHU Evaluates for large \( z \) \( z^a \, u(a, b, z) \) where \( u \) is the logarithmic confluent hypergeometric function. (double precision).
DAWS Computes Dawson’s function.
DCHU Calculates the double precision logarithmic confluent hypergeometric function.
DDAWS Computes the double precision Dawson’s function.
DPOCH Computes double precision generalized Pochhammer’s symbol.
DPOCH1 Calculates a generalization of Pochhammer’s symbol starting from first order. (double precision)
POCH Evaluates a generalization of Pochhammer’s symbol.
POCH1 Computes Pochhammer’s symbol from first order.
R9CHU Evaluates \( z^a \, CHU(a, b, z) \), for large \( z \).
C. Roots (Zeros) of Functions, Simultaneous Nonlinear Equations

**AMOSLIB**

- **FINVER** Solves \( f(x) = a \) for \( x \), \( f(x) \) monotone increasing or decreasing.
- **XNEWT** Computes an estimate of a root of a strictly monotone function.

**IMSL**

- **ZANLYT** Zeros of an analytic complex function using the Muller method with deflation.
- **ZBRENT** Zero of a function which changes sign in a given interval (Brent Algorithm).
- **ZCPOLY** Zeros of a polynomial with complex coefficients (Jenkins-Traub).
- **ZFALSE** Zero of a function, given an interval containing the zero.
- **ZPOLR** Zeros of a polynomial with real coefficients (Laguerre).
- **ZQADC** Zeros of a quadratic with complex coefficients.
- **ZQADR** Zeros of a quadratic with real coefficients.
- **ZREAL1** The real zeros of a real function – to be used when initial guesses are poor.
- **ZREAL2** The real zeros of a real function – to be used when initial guesses are good.
- **ZRPOLY** Zeros of a polynomial with real coefficients (Jenkins-Traub).
- **ZSCNT** Solve a system of nonlinear equations.
- **ZPOW** Solve a system of nonlinear equations.
- **ZSRCH** Generate starting points in an \( n \)-dimensional space, for non-linear optimization problems.

**LOCLIB**

- **CCUBIC** Finds the complex roots of a cubic polynomial with complex coefficients.
- **NS01A** Solves a system of nonlinear algebraic equations of the form \( F(x) = 0 \), where \( F \) and \( x \) are vectors of length \( n \). The function \( F \) should have continuous first derivatives, but there is no need to calculate these, because the algorithm automatically computes finite difference approximations. The method used is a compromise between Newton’s method and steepest descent. The iterative nature of the algorithm requires that an initial estimate of the solution be supplied, but this can be fairly poor and the process still will usually converge.
- **RCUBIC** Finds the roots of a general cubic polynomial with real coefficients.
**MINPACK**

**HYBRD**
Computes a numerical solution to a system of nonlinear equations by a modification of Powell's hybrid algorithm. The Jacobian matrix of partial derivatives is calculated by a forward-difference approximation or by an update formula of Broyden. This package should be used when the Jacobian matrix is not available and flexibility is required.

**HYBRD1**
Computes a numerical solution to a system of nonlinear equations by a modification of Powell's hybrid algorithm. The Jacobian matrix of partial derivatives is calculated by a forward-difference approximation or by an update formula of Broyden. This is the easy-to-use driver for HYBRD, and should be used when the Jacobian matrix is not available and flexibility is not required.

**HYBRJ**
Computes a numerical solution to a system of nonlinear equations by a modification of Powell's hybrid algorithm. The user must provide a subroutine to calculate the Jacobian matrix of partial derivatives. This package should be used when the Jacobian matrix is available and flexibility is required.

**HYBRJ1**
Computes a numerical solution to a system of nonlinear equations by a modification of Powell's hybrid algorithm. The user must provide a subroutine to calculate the Jacobian matrix of partial derivatives. This is the easy-to-use driver for HYBRJ, and should be used when the Jacobian matrix is available and flexibility is not required.

**NAG**

**C02ADF**
Finds all the roots of a complex polynomial equation, using the method of Grant and Hitchins.

**C02AEF**
Finds all the roots of a real polynomial equation, using the method of Grant and Hitchins.

**C05ADF**
Locates a zero of a continuous function in a given interval \((a,b)\) by a combination of the methods of linear interpolation, extrapolation and bisection.

**C05AGF**
Locates a simple zero of a continuous function from a given starting value, using a binary search to locate an interval containing a zero of the function, then a combination of the methods of linear interpolation, extrapolation and bisection to locate the zero precisely.

**C05AJF**
Attempts to locate a zero of a continuous function by continuation method using a secant iteration.

**C05AVF**
Attempts to locate an interval containing a simple zero of a continuous function \(f(x)\) using a binary search. It uses "reverse communication" for evaluating the function.

**C05AXF**
Attempts to locate a zero of a continuous function using a continuation method based on a secant iteration.

**C05AZF**
Locates a simple zero of a continuous function on a given interval by a combination of the methods of linear interpolation, linear extrapolation and bisection. It uses "reverse communication" for evaluating the function.

**C05NBF**
Easy-to-use routine to find a zero of a system of \(N\) non-linear functions by a modification of the Powell hybrid method. Derivatives of the function are not required.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C05NCF</td>
<td>Comprehensive version of C05NBF.</td>
</tr>
<tr>
<td>C05PBF</td>
<td>Easy-to-use routine to find a zero of a system of ( N ) non-linear functions in ( N ) variables by a modification of the Powell hybrid method. The user must provide the Jacobian.</td>
</tr>
<tr>
<td>C05PCF</td>
<td>Comprehensive version of C05PBF.</td>
</tr>
<tr>
<td>C05ZAF</td>
<td>Checks the user-provided Jacobian prior to use in C05PBF or C05PCF.</td>
</tr>
<tr>
<td>DBNDZR</td>
<td>Computes error bounds and improvements to roots of polynomials.</td>
</tr>
<tr>
<td>DC POLY</td>
<td>Finds all the roots of a polynomial with complex coefficients using double precision.</td>
</tr>
<tr>
<td>DR POLY</td>
<td>Finds all the roots of a polynomial with real coefficients using double precision.</td>
</tr>
<tr>
<td>CPOLY</td>
<td>Finds the zeros of a polynomial with complex coefficients.</td>
</tr>
<tr>
<td>DCPOLY</td>
<td>Double precision version of CPOLY.</td>
</tr>
<tr>
<td>DR POLY</td>
<td>Double precision version of RPOLY.</td>
</tr>
<tr>
<td>DZERO</td>
<td>Double precision version of ZERO.</td>
</tr>
<tr>
<td>DZONE</td>
<td>Double precision version of ZONE.</td>
</tr>
<tr>
<td>DZONEJ</td>
<td>Double precision version of ZONEJ.</td>
</tr>
<tr>
<td>R POLY</td>
<td>Finds the zeros of a polynomial with real coefficients.</td>
</tr>
<tr>
<td>ZERO</td>
<td>Finds a real root (if one exists) of a given function within an interval specified by the user. The function values at the endpoints of the specified interval must be of opposite sign. Brent’s algorithm, a combination of bisection and inverse quadratic interpolation is used.</td>
</tr>
<tr>
<td>ZONE</td>
<td>Finds a set of zeros approximately satisfying a set of ( n ) nonlinear equations, given an initial guess.</td>
</tr>
<tr>
<td>ZONEJ</td>
<td>Finds a set of zeros approximately satisfying a set of ( n ) nonlinear equations, given an initial guess and a subroutine to evaluate the Jacobian of the system. ZONEJ should be used in preference to ZONE if the user is able to furnish a subroutine to calculate the partial derivatives of the functions with respect to the variables.</td>
</tr>
<tr>
<td>CPQR79</td>
<td>Finds the zeros of a polynomial with complex coefficients.</td>
</tr>
<tr>
<td>CP ZERO</td>
<td>Finds the zeros of a polynomial with complex coefficients.</td>
</tr>
<tr>
<td>DNSQ</td>
<td>Finds 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 (Argonne) HYBRD and HYBRDJ.</td>
</tr>
<tr>
<td>DNSQE</td>
<td>The easy-to-use version of DNSQ which finds a zero of a system of ( N ) nonlinear functions in ( N ) variables by a modification of the Powell hybrid method. This code is a combination of the MINPACK codes HYBRD1 and HYBRDJ.</td>
</tr>
<tr>
<td>Category C</td>
<td>NCAR Software Catalog</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>DSOS</td>
<td>Solves a square system of nonlinear equations. Double precision version.</td>
</tr>
<tr>
<td>FZERO</td>
<td>Searches for 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.</td>
</tr>
<tr>
<td>RPQR79</td>
<td>Finds the zeros of a polynomial with real coefficients.</td>
</tr>
<tr>
<td>RPZERO</td>
<td>Finds the zeros of a polynomial with real coefficients.</td>
</tr>
<tr>
<td>SNSQ</td>
<td>Finds 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.</td>
</tr>
<tr>
<td>SNSQE</td>
<td>SNSQE is the easy-to-use version of SNSQ which finds a zero of a system of ( N ) nonlinear functions in ( N ) variables by a modification of Powell hybrid method. This code is the combination of the MINPACK codes HYBRD1 and HYBRJ1.</td>
</tr>
<tr>
<td>SOS</td>
<td>Solves a square system of nonlinear equations.</td>
</tr>
</tbody>
</table>
D2a1. Initial Value Problems for Non-Stiff Ordinary Differential Equations

IMSL

DREBS  Differential equation solver – extrapolation method.
DVERK  Differential equation solver – Runge Kutta-Verner fifth and sixth order method.

NAG

D02BAF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a Runge-Kutta-Merson method.
D02BBF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a Runge-Kutta-Merson method, and returns the solution at points specified by the user.
D02BDF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a Runge-Kutta-Merson method, and computes a global error estimate check. A stiffness check is also available.
D02BGF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a Runge-Kutta-Merson method, until a specified component attains a given value.
D02BHF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a Runge-Kutta-Merson method, until a user-specified function of the solution is zero.
D02CAF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Adams method.
D02CBF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Adams method, and returns the solution at points specified by the user.
D02CGF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Adams method, until a specified component attains a given value.
D02CHF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Adams method, until a user-specified function of the solution is zero.
D02PAF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a Runge-Kutta-Merson method. A variety of facilities for interrupting the calculation are provided. This routine is relatively complicated and is recommended only to experienced users.
D02QAF  Integrates a system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Adams method. A variety of facilities for interrupting the calculation are provided. This routine is
relatively complicated and is recommended only to experienced users.

**D02XAF**
Interpolates the solution of the system of first-order ordinary differential equations from information provided by the Runge-Kutta-Merson routine D02PAF.

**D02XBF**
Interpolates one component of the solution of a system of first-order ordinary differential equations from information provided by the Runge-Kutta-Merson routine D02PAF.

**D02XGF**
Interpolates the solution of a system of first-order ordinary differential equations from information provided by the Adams routine D02QAF or the Gear routine D02QBF.

**D02XHF**
Interpolates one component of the solution of a system of first-order ordinary differential equations from information provided by the Adams routine D02QAF or the Gear routine D02QBF.

**D02YAF**
Integrates a system of first-order ordinary differential equations over one step, using Merson’s Runge-Kutta method.

**NCARLB**

**AM1**
Primitive but easy to use package that performs one step of the implicit fourth order Adams-Moulton method for the solution of initial value problems for ordinary differential equations. This routine has no automatic error control.

**GERKN**
Special purpose package that solves the initial value problem for a system of ordinary differential equations. It uses the same Runge-Kutta process as RKFN on LOCLIB, except that a global error estimate is returned. This estimate requires more computation than that of RKFN, and is better.

**LSODE**
General-purpose package that solves the initial value problem for stiff or nonstiff system of ordinary differential equations given in explicit form. This supersedes the GEAR package on LOCLIB. While the basic methods are the same, many new capabilities have been added.

**LSODI**
Solves the initial value problem for stiff or nonstiff systems of first order ODE’s given in implicit form. It is a variant version of LSODE.

**ODEN**
Package to solve the initial value problem for a system of ordinary differential equations. It uses a variable order Adams method to minimize the number of derivative evaluations required.

**RK1**
Performs one step of the explicit fourth-order Runge-Kutta-Gill method. No error control.

**RKFN**
Package to solve the initial value problem for a system of ordinary differential equations, using Runge-Kutta-Fehlberg formulas to achieve fifth-order accuracy. This package is recommended for non-stiff or mildly stiff systems when derivative evaluations are cheap, and only low or moderate accuracy is required.

**PORT**

**DODES**
Double precision version of ODES.

**DODES1**
Double precision version of ODES1.

**DODESE**
Double precision version of ODESE.

**DODESH**
Double precision version of ODESH.
ODES
Ordinary differential equation solver that computes the solution to a set of first order ODEs given initial values for the dependent variables. Uses the modified midpoint rule with extrapolation. Only a single error tolerance may be specified for all solution components.

ODES1
Ordinary differential equation solver. Computes the solution to a set of first order ODEs given initial values for the dependent variables. ODES1 differs from ODES in providing three extra arguments to give the user more control over the convergence testing during the integration process.

ODESE
Default ERROR subprogram for PORT routine ODES1. An option is provided for using either local solution values or global in the calculation of the error criterion, and the criterion can be calculated for an actual integration step, or based on an error per unit step size.

ODESH
Default HANDLE routine for use with ODES. When ODES calls ODESH at the end of each time step, ODESH simply returns without having done anything.

SLATEC

DDEABM

DDERKF
Solves initial value problems in ordinary differential equations using a Runge-Kutta-Fehlberg scheme. Double precision version.

DEABM
Solves initial value problems in ordinary differential equations using an Adams-Bashforth method.

DERKF
Solves initial value problems in ordinary differential equations.

DSTEP2
Integrates a system of first order ordinary differential equations one step. Double precision version.

STEP2
Integrates a system of first order ordinary differential equations one step.

D2a2. Initial Value Problems for Stiff Ordinary Differential Equations

IMSL

DGEAR
Differential equation solver – variable order Adams predictor corrector method or Gears method.

LOCLIB

EPISODE
For stiff systems, a version of Gear’s method which has truly variable step size rather than step changes achieved by interpolation, as in LSODE (LOCLIB). For problems that require frequent and drastic step size changes, EPISODE may be more efficient than LSODE.

Differential and Integral Equations
GEAR
Obsolete package that performs one step using Gear's subroutine, DIFSUB. Use LSODE on LOCLIB instead.

STINT
A subroutine for the numerical solution of stiff systems of ordinary differential equations where the Jacobian matrix has complex eigenvalues near the imaginary axis.

NAG
D02EAF
Integrates a stiff system of first-order differential equations over a range with suitable initial conditions, using a variable-order variable-step Gear method.

D02EBF
Integrates a stiff system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order, variable-step Gear method, and returns the solution at points specified by the user.

D02EGF
Integrates a stiff system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order variable-step Gear method, until a specified component attains a given value.

D02EHF
Integrates a stiff system of first-order ordinary differential equations over a range with suitable initial conditions, using a variable-order, variable-step Gear method, until a user-specified function of the solution is zero.

D02QBF
Integrates a stiff system of first-order ordinary differential equations, over a range with suitable initial conditions, using a variable-order variable-step Gear method. A variety of facilities for interrupting the calculation is provided. This routine is relatively complicated and is recommended only to experienced users.

NCARLB
DIFSUB
Advances the solution of a system of ODE's one step. This routine is for stiff systems and has error control.

LSODE
General-purpose package that solves the initial value problem for stiff or nonstiff system of ordinary differential equations given in explicit form. This supersedes the GEAR package on LOCLIB. While the basic methods are the same, many new capabilities have been added.

LSODI
Solves the initial value problem for stiff or nonstiff systems of first order ODE's given in implicit form. It is a variant version of LSODE.

STFODN
Special purpose package that solves the initial value problem for a stiff system of ordinary differential equations, using collocation or implicit Runge-Kutte methods. Inefficient for non-stiff systems.

SLATEC
DDEBDF
Solves initial value problems in ordinary differential equations using backward differentiation formulae. It is intended primarily for stiff problems. Double precision version.

DEBDF
Solves initial value problems in ordinary differential equations using backward differentiation formulae. It is intended primarily for stiff problems.

DINTRP
Approximates the solution at $x$ by evaluating the polynomial computed in DSTEP2 at $x$. Must be used in conjunction with DSTEP2. Double precision version.
INTRP  
Approximates the solution at x by evaluating the polynomial computed in STEP2 at x. Must be used in conjunction with STEP2.

D2b.  
Boundary Value Problems for Ordinary Differential Equations

IMSL

DTPTB  
Solve a system of ordinary differential equations with boundary conditions at two points, using a multiple shooting method.

DVCPR  
Solve a system of ordinary differential equations with boundary conditions at two points, using a variable order, variable-step size, finite-difference method with deferred corrections.

LOCLIB

COLSYS  
Solves mixed order systems of multi-point boundary-value problems. The method of solution applies spline collocation at Gaussian points. Nonlinear problems are solved using the damped Newton's method of quasilinearization. Thus at each iteration a linearized problem is solved by collocation.

LTPBVP  
Approximates the solution of linear two-point boundary value problems. The boundary conditions may be mixed or periodic. Solves one second-order equation only.

MSHOOT  
Solves the following non-linear, two-point, boundary-value problem by multiple shooting: 
\[ \frac{dy}{dx} = f(x, y), \quad Ay(x_{init}) + By(x_{final}) = C. \]

SHOOT  
Solves a non-linear two-point boundary-value problem by ordinary shooting
\[ \frac{dy}{dx} = f(x, y), \quad y(x_{init}) = \alpha, \quad By(x_{final}) = \beta. \]

SUPORE  
Solves a linear two-point boundary-value problem with the eigenvalue parameter E, having the form 
\[ \frac{dy}{dx} = M(x, u, E)y(x) + g(x, u, E), \quad Ay(x_{init}) = \alpha, \quad By(x_{final}) = \beta \]
combined with the solution of the initial value problem 
\[ \frac{du}{dx} = f(x, u, E), \quad u(x_{init}) = \eta. \] The method of solution uses superposition coupled with an orthonormalization procedure and a variable-step integration scheme. For eigenvalue problems, this basic solution is applied iteratively until a zero (an eigenvalue) of an appropriate function is obtained.

SUPORQ  
Solves a non-linear two-point boundary-value problem of the form 
\[ \frac{dy}{dx} = f(x, y), \quad \phi(y(x_{init})) = 0, \quad \psi(y(x_{final})) = 0. \] The method of solution applies a quasilinearization process which reduces the nonlinear problem to a sequence of linear two-point boundary-value problems. In turn each linear problem is solved using superposition principles coupled with an orthonormalization procedure and a variable-step Runge-Kutta-Fehlberg integration scheme.
SUPORTN Solves linear two-point boundary problems of the form:
\[ \frac{dy}{dz} = M(z,u)y(z) + g(z,u), \]
\[ Ay(z_{init}) = \alpha, By(z_{final}) = \beta \]
coupled with the solution of the initial value problem \[ \frac{du}{dz} = f(z,u), u(z_{init}) = \eta. \] The method of solution uses superposition coupled with an orthonormalization procedure and a variable-step integration scheme.

**NAG**

D02GAF Solves the two-point boundary-value problem with assigned boundary values for a system of ordinary differential equations, using a deferred correction technique and a Newton iteration.

D02GBF Solves a general linear two-point boundary value problem for a system of ordinary differential equations using a deferred correction technique.

D02HAF Solves a nonlinear two-point boundary-value problem for a first-order system of differential equations with assigned boundary values using Merson's method and a Newton iteration in a shooting method.

D02HBF Solves the two-point boundary-value problem for a system of ordinary differential equations, using initial value techniques (D02PAF) and Newton iteration; it generalises subroutine D02HAF to include the case where parameters other than boundary values are to be determined.

D02JAF Solves a regular linear two-point boundary value problem for a single \( N(\text{th}) \) order ordinary differential equation by a Chebyshev series using collocation and least squares.

D02JBF Solves a regular linear two-point boundary value problem for a system of ordinary differential equations by Chebyshev series using collocation and least squares.

D02RAF Solves the two-point boundary-value problem with general boundary conditions for a system of ordinary differential equations, using a deferred correction technique and Newton iteration.

D02SAF Solves a two-point boundary-value problem for a system of first-order ordinary differential equations with boundary conditions, combined with additional algebraic equations. It uses initial value techniques and a modified Newton iteration in a shooting and matching method.

D02TGF Solves a system of linear ordinary differential equations by least-squares fitting of a series of Chebyshev polynomials using collocation.

**SLATEC**

BVSUP Solves a linear two-point boundary value problem using superposition coupled with an orthonormalization procedure and a variable-step integration scheme.

DBVSUP Solves a linear two-point boundary value problem using superposition coupled with an orthonormalization procedure and a variable-step integration scheme. Double precision version.
D2c.  

Eigenvalue Problems for Ordinary Differential Equations

NAG

D02KAF  
Finds a specified eigenvalue of a regular second-order Sturm-Liouville system defined on a finite range, using a Pruefer transformation and a shooting method.

D02KDF  
Finds a specified eigenvalue of a regular or singular second-order Sturm-Liouville system on a finite or infinite range, using a Pruefer transformation and a shooting method. Provision is made for discontinuities in the coefficient functions or their derivatives.

D02KEF  
Finds a specified eigenvalue of a regular singular second-order Sturm-Liouville system on a finite or infinite range, using a Pruefer transformation and a shooting method. It also reports values of the eigenfunction and its derivatives. Provision is made for discontinuities in the coefficient functions or their derivatives.

D3.  

Partial Differential Equations

IMSL

DPDES  
Solve a system of partial differential equations of the form \( \frac{\partial u}{\partial t} = f(x, t, u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}) \) using the method of lines with cubic Hermite polynomials.

LOCLIB

CROSELCI  
IFTRAN source for the FORTRAN version of file CROSEL on LOCLIB.

LIPTICCI  
IFTRAN source for the FORTRAN version of file LIPTIC on LOCLIB.

PDECOL  
A package of subroutines which is designed to solve a general parabolic system of nonlinear partial differential equations of at most second order. Both the unknown function and the independent variable may be vector-valued.

POIS  
Solves an elliptic equation with variable coefficients in one direction. This is an obsolete version of GENBUN on LOCLIB, which should be used instead.

RKFPDE  
Integrates a PDE system containing several marching equations, which may be coupled with a single elliptic equation.
**NAG**

- **D03EAF**: Solves Laplace's equation in two dimensions for an arbitrary domain bounded internally or externally by one or more closed contours, given the value of either the unknown function or its normal derivative (into the domain) at each point of the boundary.
- **D03EBF**: Uses the Strongly Implicit Procedure to calculate the solution to a system of simultaneous algebraic equations of five point molecule form on a two-dimensional topologically-rectangular mesh. (Topological means that a polar grid, for example \((r, \theta)\), can be used, being equivalent to a rectangular box.)
- **D03ECF**: Uses the Strongly Implicit Procedure to calculate the solution to a system of simultaneous algebraic equations of seven point molecule form on a three-dimensional topologically-rectangular mesh. (“Topological” means that a polar grid, for example, can be used if it is equivalent to a rectangular box.)
- **D03PAF**: Integrates a single linear or non-linear parabolic partial differential equation in one space variable, using the method of lines and Gear's method.
- **D03PBF**: Integrates a system of linear or nonlinear parabolic partial differential equations in one space variable, using the method of lines and Gear's method.
- **D03PGF**: Integrates a system of nonlinear parabolic partial differential equations in one space variable, using the method of lines and Gear's method. This routine provides quite general facilities; for simpler versions see D03PAF (for a single equation), or D03PBF (for simple systems).
- **D03UAF**: Performs at each call one iteration of the Strongly Implicit Procedure. It is used to calculate on successive calls a sequence of approximate corrections to the current estimate of the solution when solving a system of simultaneous algebraic equations for which the iterative update matrix is of five point molecule form on a two-dimensional topologically-rectangular mesh. (Topological means that a polar grid, for example \((r, \theta)\), can be used, being equivalent to a rectangular box.)
- **D03UBF**: Performs at each call one iteration of the Strongly Implicit Procedure. It is used to calculate on successive calls the sequence of approximate corrections to the solution when solving a system of simultaneous algebraic equations for which the iterative update matrix is of seven-point molecule form on a three-dimensional topologically-rectangular mesh. (Topological means that a polar grid, for example, can be used if it is equivalent to a rectangular box.)

**NCARLB**

- **BLKTRI**: Solves block tridiagonal linear systems that arise from finite difference approximations to separable two-dimensional elliptic partial differential equations.
- **CBLKTRI**: Solves complex block tridiagonal linear systems that arise from finite difference approximations to separable, complex, two-dimensional, elliptic, partial differential equations.
- **CMGNBN**: Solves a complex block tridiagonal linear system of equations arising from finite difference approximations to separable, complex, two-dimensional, elliptic, partial differential equations with constant coefficients in one direction.
- **CROSEL**: Computes the second-order finite difference approximation or a fourth-order approximation to the solution of a non-separable elliptic partial differential equation with a cross derivative term on a rectangle. Boundary conditions allowed include periodic, specified (Dirichlet), or fully mixed (oblique derivatives at...
GENBUN Solves the linear system of equations resulting from a finite difference approximation on a centered grid to certain two-dimensional, elliptic, partial differential equations with constant coefficients in one direction.

HSTCRT Solves the standard five-point finite difference approximation on a staggered grid to the two-dimensional Helmholtz equation in Cartesian coordinates.

HSTCSP Solves the standard five-point finite difference approximation on a staggered grid to a two-dimensional, modified Helmholtz equation in cylindrical coordinates, assuming axisymmetry (no dependence on longitude).

HSTCYL Solves the standard five-point finite difference approximation on a staggered grid to a two-dimensional, modified Helmholtz equation in cylindrical coordinates.

HSTPLR Solves the standard five-point finite difference approximation on a staggered grid to the two-dimensional Helmholtz equation in polar coordinates.

HSTSSP Solves the standard five-point finite difference approximation on a staggered grid to the two-dimensional Helmholtz equation in spherical coordinates and on the surface of the unit sphere.

HW3CRT Solves the standard seven-point finite difference approximation on a centered grid to the three-dimensional Helmholtz equation in Cartesian coordinates.

HWSCRT Solves the standard five-point finite difference approximation on a centered grid to the two-dimensional Helmholtz equation in Cartesian coordinates.

HWSCSP Solves a finite difference approximation on a centered grid to a two-dimensional, modified Helmholtz equation in spherical coordinates, assuming axisymmetry (no dependence on longitude).

HWSCYL Solves a finite difference approximation on a centered grid to a two-dimensional, modified Helmholtz equation in cylindrical coordinates.

HWSPLR Solves the finite difference approximation on a centered grid to the two-dimensional Helmholtz equation in polar coordinates.

HWSSSP Solves a finite difference approximation on a centered grid to a two-dimensional Helmholtz equation in spherical coordinates and on the surface of the unit sphere.

LIPTIC A package for solving linear two-dimensional elliptic partial differential equations on a rectangle. The PDE is automatically discretized and any combination of periodic, specified, or mixed boundary conditions can be incorporated. There is an option for calculating a fourth-order approximation in addition to the usual second-order finite difference approximation.

POIS3D Solves a block tridiagonal, linear system of equations that arises from finite difference approximations to three-dimensional elliptic partial differential equations in a box.

POISTG Solves a block tridiagonal, linear system of equations that arises from finite difference approximations on a staggered grid to two-dimensional elliptic partial differential equations with constant coefficients in one direction.

SEPELI Automatically discretizes and solves second-order finite difference approximation and (optionally) fourth-order approximation on a uniform grid to a separable, elliptic, partial differential equation on a rectangle. Mixed or periodic boundary conditions are permitted.

SEPX4 Solves second- and (optionally) fourth-order finite difference approximation on a uniform grid to certain, separable, elliptic, partial differential equations. The
equations solved by this package are a special case of the separable equations solved by the package SEPELI, but when applicable it is at least three times faster than SEPELI.

**SLATEC**

- **BLKTRI** Solves a block tridiagonal system of linear equations (usually resulting from the discretization of separable two-dimensional elliptic equations).
- **CBLKTR** Solves a block tridiagonal system of linear equations (usually resulting from the discretization of separable elliptic equations). It is a complex version of subroutine BLKTRI.
- **CMGNBN** Solves a complex block tridiagonal linear system of equations by a cyclic reduction algorithm.
- **GENBUN** Solves by a cyclic reduction algorithm the linear system of equations that results from a finite difference approximation to certain two-dimensional elliptic PDE's on a centered grid.
- **HSTCRT** Solves the standard five-point finite difference approximation on a staggered grid to the Helmholtz equation in Cartesian coordinates.
- **HSTCSP** Solves 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** Solves the standard five-point finite difference approximation on a staggered grid to the modified Helmholtz equation in cylindrical coordinates.
- **HSTPLR** Solves a finite difference approximation to the Helmholtz equation in polar coordinates.
- **HW3CRT** Solves the standard seven-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
- **HWSCRT** Solves the standard five-point finite difference approximation to the Helmholtz equation in Cartesian coordinates.
- **HWSCSP** Solves a finite difference approximation to the modified Helmholtz equation in spherical coordinates assuming axisymmetry (no dependence on longitude).
- **HWSCYL** Solves a standard finite difference approximation to the Helmholtz equation in cylindrical coordinates.
- **HWSPLR** Solves a finite difference approximation to the Helmholtz equation in polar coordinates.
- **HWSSSP** Solves a finite difference approximation to the Helmholtz equation in spherical coordinates and on the surface of the unit sphere (radius of 1).
- **POIS3D** This subroutine solves three-dimensional block tridiagonal linear systems arising from finite difference approximations to three-dimensional Poisson equations using the Fourier transform package FFTPAK.
- **POISTG** Solves a block tridiagonal system of linear equations that results from a staggered grid finite difference approximation to two-dimensional elliptic PDE's.
SEPELI  Automatically discretizes and solves second- and (optionally) fourth-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  Solves 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.

D5. Integral Equations

NAG

D05AAF  Solves a linear, non-singular Fredholm equation of the second kind with a split kernel.

D05ABF  Solves any linear non-singular Fredholm integral equation of the second kind with a smooth kernel.

D7. Conformal Mapping, Schwarz-Christoffel Transformations

LOCLIB

SCPACK  Computes Schwarz-Christoffel transformations (and their inverses) that map the unit disk conformally onto the interior of a bounded or unbounded polygon in the complex plane. Such transformations may be applied to solve the Laplace and Poisson equations and related problems in two-dimensional domains with irregular or unbounded (but not curved or multiply connected) geometries. The time required to solve the mapping problem is roughly proportional to $n^3$, where $n$ is the number of vertices of the polygon.
E1d. Interpolation of a Function of One Variable

FITPACK

CURV1 Determines the parameters necessary to compute an interpolatory spline under tension through a sequence of functional values. The slopes at the two ends of the curve may be specified or omitted. For actual computation of points on the curve it is necessary to call the function CURV2.

CURVN1 Determines the parameters necessary to compute a natural interpolatory spline under tension through a sequence of functional values. For actual computation of points on the curve it is necessary to call the function CURV2.

CURVP1 Determines the parameters necessary to compute a periodic interpolatory spline under tension through a sequence of functional values. For actual ends of the curve may be specified or omitted. For actual computation of points on the curve it is necessary to call the function CURVP2.

CURVPS Determines the parameters necessary to compute a periodic smoothing spline under tension. For a given increasing sequence of abscissae $x_i, i=1, \ldots, n$ and associated ordinates $y_i, i=1, \ldots, n$, letting $P$ be the period, $x_{n+1}=x_1+P$, and $y_{n+1}=y_1$, the function determined minimizes

$$
\sum_{i=1}^{n} f''(x_i)^2 + \sigma^2 \left( \frac{f'(x_{i+1}) - f'(x_i)}{x_{i+1} - x_i} \right)^2
$$

over all functions $f$ with period $P$ and two continuous derivatives such that $\sum_{i=1}^{n} \left( \frac{f(x_i) - y_i}{d_i} \right)^2 \leq S$, where $S$ is a given constant and where $d_i, i=1, \ldots, n$ are a given set of observation weights. The function determined is a periodic spline under tension with third derivative discontinuities at $x_i, i=2, \ldots, n$ (and all periodic translations of these values). For actual computation of points on the curve it is necessary to call the function CURVP2.

CURVS Determines the parameters necessary to compute a smoothing spline under tension. For a given increasing sequence of abscissae $x_i, i=1, \ldots, n$ and associated ordinates $y_i, i=1, \ldots, n$, the function determined minimizes

$$
\sum_{i=1}^{n-1} f''(x_i)^2 + \sigma^2 \left( \frac{f'(x_{i+1}) - f'(x_i)}{x_{i+1} - x_i} \right)^2
$$

over all functions $f$ with two continuous derivatives such that $\sum_{i=1}^{n} \left( \frac{f(x_i) - y_i}{d_i} \right)^2 \leq S$, where $S$ is a given constant and where $d_i, i=1, \ldots, n$ are a given set of observation weights. The function determined is a spline under tension with third derivative discontinuities at $x_i, i=2, \ldots, n-1$. For actual computation of points on the curve it is necessary to call the function CURV2.

KURV1 Determines the parameters necessary to compute an interpolatory spline under tension forming a curve in the plane and passing through a sequence of pairs $(x_1, y_1), \ldots, (x_n, y_n)$. For actual computation of points on the curve it is necessary to call the function KURV2.
KURVN1 Determine the parameters necessary to compute a natural interpolatory spline under tension forming a curve in the plane and passing through a sequence of pairs \((x_1, y_1), \ldots, (x_n, y_n)\). For actual computation of points on the curve it is necessary to call the function KURV2.

KURVP1 Determine the parameters necessary to compute a periodic interpolatory spline under tension forming a curve in the plane and passing through a sequence of pairs \((x_1, y_1), \ldots, (x_n, y_n)\). For actual computation of points on the curve it is necessary to call the function KURVP2.

QURV1 Determine the parameters necessary to compute an interpolatory spline under tension passing through a sequence of triples \((x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\). The slopes at the two ends of the curve may be specified or omitted. For actual computation of points on the curve it is necessary to call the function QURV2.

QURVN1 Determine the parameters necessary to compute a natural interpolatory spline under tension passing through a sequence of triples \((x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\). For actual computation of points on the curve it is necessary to call the function QURV2.

QURVP1 Determine the parameters necessary to compute an interpolatory spline under tension forming a closed curve in three-dimensional space passing through a sequence of triples \((x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\). For actual computation of points on the curve it is necessary to call the function QURVP2.

IMSL

ICSCCU Cubic spline interpolation (easy-to-use version).
ICSICU Interpolatory approximation by cubic splines with arbitrary second derivative end conditions.
ICSPLN Cubic spline interpolation with periodic end conditions.
IQHSCU One-dimensional quasi-cubic Hermite interpolation.

NAG

E01AAF Interpolates at a given point \(X\) from a table of function values \(y_i\) evaluated at equidistant or non-equidistant points \(x_i\) \((i=1,2,\ldots,N+1)\), using Aitken’s technique of successive linear interpolations.
E01ABF Interpolates at a given point \(X\) from a table of function values evaluated at equidistant points, by Everett’s formula.
E01AEF Constructs the Chebyshev-series representation of a polynomial interpolant to a set of data which may contain derivative values.
E01BAF Determines a cubic-spline interpolant to a given set of data.
E01RAF Computes a rational interpolation function (represented as a continued fraction) for a set of data.
E01RBF Evaluates a rational function computed by E01RAF.
E02AFF Computes the coefficients of a polynomial, in its Chebyshev-series form, which interpolates (passes exactly through) data at a special set of points. Least-squares polynomial approximations can also be obtained.
NCAR Software Catalog

NCARLB

BSL1NT Performs one-dimensional local linear, quadratic, or cubic interpolation (using two, three, or four adjacent points, respectively) for values and derivatives. The subroutine is efficient, but the resulting curve has no continuous derivatives.

CUBSPL Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives.

HRM1NT Performs local linear, cubic, or quintic fitting (using two adjacent values, two adjacent values plus first derivative values, or two adjacent values plus first and second derivative values, respectively). The resulting curve has either zero, one, or two continuous derivatives, respectively, but requires this derivative information in all but the linear case.

PORT

CSPFI Fits a cubic spline function to input data pairs \((x, y)\). Most users will find it easier to use CSPIN (interpolation), CSPDI (differentiation), or CSPQU (integration), each of which calls CSPFI.

CSPIN Interpolates at requested points in given input data using a spline approximation.

DCSPFI Double precision version of CSPFI.

DCSPIN Double precision version of CSPIN.

SLATEC

BINT4 Computes the \(B\) representation of a cubic spline which interpolates data \((x_i, y_i)\), \(i = 1, \ldots, n\).

BINTK Produces the \(B\)-spline coefficients, BCOEF, of the \(B\)-spline of order \(k\) with knots \(t_i, i = 1, \ldots, n + k\), which takes on the value \(y_i\) at \(x_i, i = 1, \ldots, n\).

DBINT4 Computes the \(B\)-representation of a cubic spline which interpolates data \((x_i, y_i)\), \(i = 1, \ldots, n\). Double precision version.

DBINTK Produces the \(B\)-spline coefficients, BCOEF, of the \(B\)-spline of order \(k\) with knots \(t_i, i = 1, \ldots, n + k\), which takes on the value \(y_i\) at \(x_i, i = 1, \ldots, n\). Double precision version.

PCHIA Computes the definite integral of a piecewise cubic Hermite interpolant over an arbitrary interval.

PCHIC Determines values for the internal derivatives to guarantee a monotone piecewise cubic Hermite interpolant to monotone data. This is used if neither of the above conditions holds, or if the user wishes control over boundary derivatives and treatment of the internal derivatives at points where the monotonicity of the data changes. PCHIC will generally reproduce monotonicity on subintervals over which the data are monotonic.

PCHID Computes the definite integral of a piecewise cubic Hermite interpolant over an interval whose end points are data points.

PCHIM Determines values for the internal derivatives to guarantee a monotone piecewise cubic Hermite interpolant to monotone data. This is used if the data are monotonic or if the user wants to guarantee that the interpolant stays within the limits of the data, even when the data is not monotonic.
PCHSP  Determines the cubic spline interpolant to given data. User has control over boundary conditions.

POLINT  To produce the polynomial which interpolates a set of discrete data points.

POLYVL  Calculates the value of the polynomial and its first NDER derivatives where the polynomial was produced by a previous call to POLINT.

PVALUE  To 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.

E1e.  Interpolation of a Function of Two or More Variables

FITPACK

BSURF1  Determines the parameters necessary to compute an interpolatory surface passing through a rectangular grid of functional values. The x and y values are assumed equally spaced in the grid. The surface determined can be represented as the tensor product of splines under tension. For actual interpolation at a grid of points equally spaced in both x and y coordinates it is necessary to call subroutine BSURF2.

NBTOH2  Obtains a Hermite interpolation representation (i.e. function values and second partial derivatives) for a three-dimensional spline from a B-spline basis representation. The spline is assumed to have natural end conditions.

NBTOH3  Obtains a Hermite interpolation representation (i.e. function values and partial derivatives) for a three-dimensional spline from a B-spline basis representation. The spline is assumed to have natural end conditions.

NSURF1  Determines the parameters necessary to compute an interpolatory surface passing through a rectangular grid of functional values. The surface determined can be represented as the tensor product of splines under tension. For actual mapping of points onto the surface it is necessary to call the function SURF2. For actual interpolation at a grid of points equally spaced in both x and y coordinates it is necessary to call subroutine NSURF2.

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

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

SURFB1  Determines the parameters necessary to compute an interpolatory function on a two-dimensional rectangular grid. The function determined can be represented as the tensor product of splines under tension. For actual mapping of points it is necessary to call the function SURFBD, which also returns first and second
partial derivatives.

SURFN1 Determines the parameters necessary to compute an interpolatory surface passing through a rectangular grid of functional values. The surface determined can be represented as the tensor product of natural splines under tension. For actual mapping of points onto the surface it is necessary to call the function SURF2.

VAL3B1 Determines the parameters necessary to compute an interpolatory function on a three-dimensional rectangular grid. The function determined can be represented as a tensor product of splines under tension. For actual mapping of points it is necessary to call the subroutine VAL3BD, which also returns first and second partial derivatives.

XSURF1 Determines the parameters necessary to compute an interpolatory surface passing through a rectangular grid of functional values. The $x$ values are assumed equally spaced in the grid. The surface determined can be represented as the tensor product of splines under tension. For actual interpolation at a grid of points equally spaced in both $x$ and $y$ coordinates it is necessary to call subroutine XSURF2.

YSURF1 Determines the parameters necessary to compute an interpolatory surface passing through a rectangular grid of functional values. The $y$ values are assumed equally spaced in the grid. The surface determined can be represented as the tensor product of splines under tension. For actual interpolation at a grid of points equally spaced in both $x$ and $y$ coordinates it is necessary to call subroutine YSURF2.

ZURFN1 Determines the parameters necessary to compute an interpolatory surface passing through a grid of points in three-dimensional space. The surface determined can be represented as three tensor product of natural splines under tension. For actual mapping of points onto the surface it is necessary to call the function ZURF2D, which also returns derivatives with respect to two parameters for use in calculating normal vectors, tangent planes, curvatures, etc.

IMSL

IBCCCU Bicubic spline two-dimensional coefficient calculator.
IBCIEU Bicubic spline two-dimensional interpolator.

LOCLIB

BILIN Performs a bilinear interpolation of data on a two-dimensional rectangular grid.
CTRINT Produces two-dimensional arrays of function values from contour maps digitized by the Bendix Datagrid equipment. The data must first be processed by the prepass program in PREPA.
TRIANGLE Performs two-dimensional linear interpolation over an arbitrarily located set of points in the plane.
TRIVAR Interpolates functions of three variables.
**NAG**

**E01ACF**
Interpolates at a given point \((A, B)\) from a table of function values defined on a rectangular grid in the X-Y plane, by fitting bi-cubic spline functions.

**NCARLB**

**BIVAR**
Provides bivariate interpolation and smooth surface fitting for irregularly distributed data.

**BSL2NT**
Performs two-dimensional local interpolation by finding a surface which, on each rectangular piece, is the product of a polynomial in one variable times a polynomial in the other variable. The polynomials can be linear, quadratic, or cubic, but the resulting surface has no continuous derivatives.

**CUBSPL**
Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives.

**QSHEP**
Performs interpolation of irregularly distributed data as a function of two variables.

---

**E2a. Linear Least Squares Approximation, Overdetermined Linear Systems (see also L08a3)**

**IMSL**

**ICSFKU**
Least squares approximation by cubic splines – fixed knots.

**ICSVKU**
Least squares approximation by cubic splines – variable knots.

**IFLSQ**
Least squares approximation with user supplied functions.

**IQHSCV**
Smooth surface fitting with irregularly distributed data points.

**LLBQF**
High accuracy solution of linear least squares problem.

**LLSQF**
Solution of a linear least squares problem.

**OFIMA3**
Least squares solution to the matrix equation \(A t = b\).

**LOCLIB**

**FITEDIT**
Detects wild data values by fitting polynomials of degree less than or equal to 2 by the method of least squares.

**LSPACK**
Solves both unconstrained and constrained least-squares problems and computes the associated covariance matrices. These routines are from Hanson and Lawson's book *Solving Least-Squares Problems*.

**SOLMLS**
Solves the minimal least squares problem; that is, it calculates a vector \(x\) which has minimum Euclidean length among all the vectors such that the Euclidean length of the residual does not exceed a user specified value. This package is recommended for problems that are poorly conditioned.
NAG

E02ADF Computes weighted least-squares polynomial approximations to an arbitrary set of data points.

E02AFF Computes the coefficients of a polynomial, in its Chebyshev-series form, which interpolates (passes exactly through) data at a special set of points. Least-squares polynomial approximations can also be obtained.

E02AGF Computes constrained weighted least-squares polynomial approximations in Chebyshev-series form to an arbitrary set of data points. The values of the approximations and any number of their derivatives can be specified at selected points.

E02BAF Computes a weighted least-squares approximation to an arbitrary set of data points by a cubic spline with knots prescribed by the user. Cubic spline interpolation can also be carried out.

E02CAF Forms an approximation to the weighted, least-squares Chebyshev series surface fit to data arbitrarily distributed on lines parallel to one independent co-ordinate axis.

E02DAF Forms a minimal, weighted least-squares bicubic spline surface fit with prescribed knots to a given set of data points.

F01AXF Reduces an \( m \times n \) real matrix, \( m \geq n \), to upper triangular form for use in F04AMF and F04ANF. The routine uses Householder transformations with column pivoting.

F04AMF Calculates the accurate least squares solution of a set of \( M \) linear equations in \( n \) unknowns, \( M \geq N \) and rank = \( N \) with multiple right hand sides, \( AX = B \).

F04ANF Calculates the approximate least squares solution of a set of \( M \) linear equations in \( n \) unknowns, \( M \geq N \) and rank = \( N \) with a single right hand side, \( Ax = b \), where \( A \) has been decomposed into triangular matrices using F01AXF.

F04JAF Finds the minimal solution of a linear least squares problem, \( Ax = b \), where \( A \) is a real \( m \times n \) (\( m \geq n \)) matrix and \( b \) is an \( m \) element vector.

F04JDF Finds the minimal solution of a linear least squares problem, \( Ax = b \), where \( A \) is a real \( m \times n \) (\( m \leq n \)) matrix and \( b \) is an \( m \) element vector.

F04JGF Finds the solution of a linear least squares problem, \( Ax = b \), where \( A \) is a real \( m \times n \) (\( m \geq n \)) matrix and \( b \) is an \( m \) element vector. If the matrix of observations is not of full rank, then the minimal least squares solution is returned.

F04QAF Solves sparse unsymmetric equations, sparse linear least squares problems and sparse damped linear least squares problems using a Lanczos algorithm.

NCARLB

HSHSLV Solves the linear least squares problem for a real, (possibly) overdetermined linear system, using Householder transformations.

LSPOLY Computes the coefficients of a low-degree polynomial that is the best least squares fit to a given set of data.

SPLPAK Performs least squares fitting of multi-dimensional cubic splines to arbitrarily located data.
SUPRLS Determines the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.

SVDSLV Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.

PORT

CLST2 Finds the least-squares solution of a complex linear algebraic system of equations, $AX = B$, where $A$ is $M$ times $N$ ($M \geq N$), $X$ is $N$ times $NB$, and $B$ is $M$ times $NB$. A QR decomposition of the coefficient matrix using Householder transformations is followed by a back substitution for the solutions.

DCLST2 Double precision version of CLST2.

DDL2SF Double precision version of DL2SF.

DDL2SW Double precision version of DL2SW.

DL2SF Fits discrete data with a B-spline expansion, using basis B-splines of a specified order and a least squares fit to obtain the expansion coefficients.

DL2SFF Double precision version of L2SFF.

DL2SFH Double precision version of L2SFH.

DL2SW Fits discrete data with a B-spline expansion, using basis B-splines of a specified order and a weighted least squares fit to obtain the expansion coefficients.

DLSTSQ Double precision version of LSTSQ.

L2SFF Fits a function with B-splines. Obtains an expansion of a known function in terms of basis B-splines of a specified order, using a weighted least-squares fit to obtain the coefficients of the expansion.

L2SFH Fits B-splines to the derivatives of a function. Obtains a B-spline approximation to a function and its derivatives through a specified order, by minimizing a functional involving least-squares integrals on a given mesh. The minimization determines coefficients in an expansion of the function in terms of B-splines of a specified order.

LSTSQ Finds the least-squares solution of a system of linear equations, where the problem can be solved for several right-hand sides with one call. A QR decomposition of the coefficient matrix using Householder transformations is followed by a back substitution for the solution.

SLATEC

BNDACC Introduces new blocks of data for banded least square problems. See BNDSOL.

BNDSOL Solve the least squares problem $Ax = b$ for banded matrices $A$ using sequential accumulation of rows of the data matrix. Exactly one right-handed side vector is permitted.

CQRSL Applies the output of CQRDC to compute coordinate transformations, projections, and least squares solutions.

DQRSL Applies the output of DQRDC to compute coordinate transformations, projections, and least squares solutions. Double precision version.
EFC | Fits 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 | Fits 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.

HFTI | Solves the least squares problem $Ax = b$ for banded matrices $A$ using sequential accumulation of rows of the data matrix. Exactly one right-handed side vector is permitted.

LLSIA | Solves linear least squares problems by performing a QR factorization of the matrix $A$ using Householder transformations. Emphasis is put on detecting possible rank deficiency.

LSEI | Solve a linearly constrained least squares problem with equality and inequality constraints, and optionally compute a covariance matrix.

MINFIT | Computes singular value decomposition of rectangular matrix and solve related linear least squares problem.

POLFIT | To fit data in a least squares sense by polynomials in one variable.

SGLSS | Solves linear least squares problems by performing a QR factorization of the matrix $A$ using Householder transformations. Emphasis is put on detecting possible rank deficiency.

WNNSLS | Solve a linearly constrained least squares problem with equality constraints and nonnegativity constraints on selected variables.

E2b. | **Nonlinear Least Squares Approximation (see also L08b)**

**IMSL**

ZXSSQ | Minimum of the sum of squares of $m$ functions in $n$ variables using a finite difference Levenberg-Marquardt Algorithm.

**LOCLIB**

NL2SOL | Solves nonlinear least squares problems. That is, given a $p$-vector $x$ of parameters and a user-provided routine that computes an $n$-vector $r = r(x)$ of residuals corresponding to $x$ ($r(x)$ probably arises from a nonlinear model involving $p$ parameters and $n$ observations) NL2SOL seeks a parameter vector $x$ that minimizes the sum of the squares of (the components of) $r(x)$. Includes several novel features which, in the large residual case, make it more reliable than the Levenberg-Marquardt method, and more efficient than secant or variable metric methods, which are intended for general function minimization.
MINPACK

LMDER
Minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The user must provide a subroutine to calculate the Jacobian matrix of partial derivatives. This package should be used when the Jacobian matrix is available, flexibility is required, and storage space for the Jacobian is not a problem.

LMDER1
Minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The user must provide a subroutine to calculate the Jacobian matrix of partial derivatives. This is the easy-to-use driver for LMDER, and should be used when the Jacobian matrix is available, flexibility is not required, and storage space for the Jacobian is not a problem.

LMDIF
Minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The Jacobian matrix of partial derivatives is calculated by a forward-difference approximation. This package should be used when the Jacobian matrix is not available and flexibility is required.

LMDIF1
Minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The Jacobian matrix of partial derivatives is calculated by a forward-difference approximation. This is the easy-to-use driver for LMDIF, and should be used when the Jacobian matrix is not available and flexibility is not required.

LMSTR
Minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The user must provide a subroutine to calculate the Jacobian matrix of partial derivatives, one row per call. This package should be used when the Jacobian matrix is available, flexibility is required, and storage space is limited.

LMSTR1
Minimizes the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The user must provide a subroutine to calculate the Jacobian matrix of partial derivatives, one row per call. This is the easy-to-use driver for LMSTR, and should be used when the Jacobian matrix is available, flexibility is not required, and storage space is limited.

NAG

E04FCF
Is a comprehensive algorithm for finding an unconstrained minimum of a sum of squares of M nonlinear functions in N variables (M ≥ N). No derivatives are required.

E04FDF
Is an easy-to-use algorithm for finding an unconstrained minimum of a sum of squares of M nonlinear functions in N variables (M ≥ N). No derivatives are required.

E04GBF
Is a comprehensive quasi-Newton algorithm for finding an unconstrained minimum of a sum of squares of M non-linear functions in N variables (M ≥ N). First derivatives are required.

E04GCF
Is an easy-to-use quasi-Newton algorithm for finding an unconstrained minimum of a sum of squares of M nonlinear functions in N variables (M ≥ N). First derivatives are required. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work
even if the derivatives have occasional discontinuities).

**E04GDF**
Is a comprehensive modified Gauss-Newton algorithm for finding an unconstrained minimum of a sum of squares of \( M \) non-linear functions in \( N \) variables \((M \geq N)\). First derivatives are required.

**E04GEF**
Is an easy-to-use modified Gauss-Newton algorithm for finding an unconstrained minimum of a sum of squares of \( M \) nonlinear functions in \( N \) variables \((M \geq N)\). First derivatives are required.

**E04HEF**
Is a comprehensive modified Gauss-Newton algorithm for finding an unconstrained minimum of a sum of squares of \( M \) non-linear functions in \( N \) variables \((M \geq N)\). First and second derivatives are required.

**E04HFF**
Is an easy-to-use modified Gauss-Newton algorithm for finding an unconstrained minimum of a sum of squares of \( M \) nonlinear functions in \( N \) variables \((M \geq N)\). First and second derivatives are required.

**NCARLB**

**NLLSSQ**
Performs a least squares fit of a given nonlinear parameterized function to given data.

**SLATEC**

**SCOV**
Calculates 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**
Minimizes 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**
Easy-to-use version of SNLS1 which minimizes the sum of the squares of \( m \) nonlinear functions in \( n \) variables by a modification of the Levenberg-Marquardt algorithm. This code is the combination of the MINPACK codes LMDER1, LMDIF1, and LMSTR.

**E5.**
Minimax (L-Infinity), L1, and Other Approximations, Over-determined Linear Systems

**IMSL**

**ICSMOU**
One-dimensional data smoothing by error detection.

**ICSSCU**
Cubic spline data smoother.

**ICSSCV**
Cubic spline data smoother (easy-to-use version).

**IRATCU**
Rational weighted Chebyshev approximation of a continuous function.

**RLLMV**
Perform linear regression using the minimax criterion.
NAG

E02ACF  Calculates a minimax polynomial fit to a set of data points.
E02GAF  Calculates an $L_1$ solution to an over-determined system of linear equations.
E02GBF  Calculates an $L_1$ solution to an over-determined system of linear equations, possibly subject to linear inequality constraints.
E02GCF  Calculates an $L_{\infty}$ solution to an over-determined system of linear equations.
E02RAF  Calculates the coefficients in a Padé approximant to a function from its user-supplied Maclaurin expansion.

PORT

BURAM  Finds the best uniform rational approximation to a given function on a specified mesh.
BURM1  Finds the best uniform rational approximation to a given function on a specified mesh, starting from a given initial approximation. If a good starting approximation is not available, use BURAM instead.
DBURAM  Double precision version of BURAM.
DBURM1  Double precision version of BURM1.

E8a. Evaluation of Interpolating or Approximating Functions using B-Splines

FITPACK

BSURF2  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine BSURF1 should be called earlier to determine certain necessary parameters. In both BSURF1 and BSURF2, the original grid is assumed to be equally spaced in the $x$ and $y$ coordinates.

BSURF3  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine BSURF1 should be called earlier to determine certain necessary parameters. In both BSURF1 and BSURF3, the original grid is assumed to be equally spaced in the $x$ and $y$ coordinates. The surface values along horizontal lines are output one by one (starting with the line with largest $y$-coordinate) by calling a user provided subroutine.

NSURF2  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine SURF1 or NSURF1 should be called earlier to determine certain necessary parameters.

NSURF3  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension.
tension. The subroutine SURF1 or NSURF1 should be called earlier to determine certain necessary parameters. The surface values along horizontal lines are output one by one (starting with the line with largest \( y \)-coordinate) by calling a user provided subroutine.

**XSURF2**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a \( B \)-spline under tension. The subroutine XSURF1 should be called earlier to determine certain necessary parameters. In both XSURF1 and XSURF2, the original grid is assumed to be equally spaced in the \( x \) coordinate.

**XSURF3**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a \( B \)-spline under tension. The subroutine XSURF1 should be called earlier to determine certain necessary parameters. In both XSURF1 and XSURF3, the original grid is assumed to be equally spaced in the \( x \) coordinate. The surface values along horizontal lines are output one by one (starting with the line with largest \( y \)-coordinate) by calling a user provided subroutine.

**YSURF2**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a \( B \)-spline under tension. The subroutine YSURF1 should be called earlier to determine certain necessary parameters. In both YSURF1 and YSURF2, the original grid is assumed to be equally spaced in the \( y \) coordinate.

**YSURF3**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a \( B \)-spline under tension. The subroutine YSURF1 should be called earlier to determine certain necessary parameters. In both YSURF1 and YSURF3, the original grid is assumed to be equally spaced in the \( y \) coordinate. The surface values along horizontal lines are output one by one (starting with the line with largest \( y \)-coordinate) by calling a user provided subroutine.

**NAG**

**E02BBF**
Evaluates a cubic spline from its \( B \)-spline representation.

**E02BCF**
Evaluates a cubic spline and its first three derivatives from its \( B \)-spline representation.

**E02DBF**
 Calculates values of a bicubic spline from its \( B \)-spline representation.

**PORT**

**BSPL1**
Evaluates, at a given set of points in a specified mesh interval, basis splines together with selected orders of derivatives.

**BSPLD**
Evaluates basis splines and their derivatives at a given set of points in a specified mesh interval.

**BSPLN**
Evaluates, at a given set of points in a specified mesh interval, all the basis splines which are nonzero in that interval.

**DBSPL1**
Double precision version of BSPL1.

**DBSPLD**
Double precision version of BSPLD.

**DBSPLN**
Double precision version of BSPLN.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEEBSF</td>
<td>Double precision version of EEBSF.</td>
</tr>
<tr>
<td>DEEBSI</td>
<td>Double precision version of EEBSI.</td>
</tr>
<tr>
<td>DEESFF</td>
<td>Double precision version of EESFF.</td>
</tr>
<tr>
<td>DEESFI</td>
<td>Double precision version of EESFI.</td>
</tr>
<tr>
<td>DSPLN1</td>
<td>Double precision version of SPLN1.</td>
</tr>
<tr>
<td>DSPLN2</td>
<td>Double precision version of SPLN2.</td>
</tr>
<tr>
<td>DSPLND</td>
<td>Double precision version of SPLND.</td>
</tr>
<tr>
<td>DSPLNE</td>
<td>Double precision version of SPLNE.</td>
</tr>
<tr>
<td>EEBSF</td>
<td>Estimates the error in a given B-spline fit to a function by refining the mesh.</td>
</tr>
<tr>
<td>EEBSI</td>
<td>Estimates the error in a given B-spline fit to a function by refining the mesh. The error estimation is done over a set of $x$ intervals selected by the user.</td>
</tr>
<tr>
<td>EESFF</td>
<td>Finds the maximum absolute error in a given B-spline fit to a function.</td>
</tr>
<tr>
<td>EESFI</td>
<td>Finds the maximum absolute error in a given B-spline fit to a function over a set of $x$ intervals specified by the user.</td>
</tr>
<tr>
<td>SPLN1</td>
<td>Spline and selected derivatives evaluation. Evaluates a function described by a (previously determined) expansion in terms of B-splines, and also uses the expansion to evaluate selected orders of derivatives. The orders of the derivatives must be less than the order of the B-splines used in the approximation. If many calls to SPLN1 are anticipated, only differing in choices for the points of evaluation, it would be cheaper to use SPLN2, since SPLN2 saves intermediate results from one call to the next.</td>
</tr>
<tr>
<td>SPLN2</td>
<td>Tuned spline evaluation. Evaluates a function described by a (previously determined) expansion in terms of B-splines, and also uses the expansion to evaluate selected orders of derivatives. The orders of the derivatives must be less than the order of the B-splines used in the approximation. SPLN2 differs from SPLN1 in providing four additional parameters which can be set by the user to achieve greater efficiency.</td>
</tr>
<tr>
<td>SPLND</td>
<td>Spline and derivatives evaluation. Evaluates, at a given set of points, a function described by a (previously determined) expansion in terms of B-splines. Derivatives of the sum (evaluated at the same given points) through a specified order are also returned. If many calls with a small number of points of evaluation to SPLND are anticipated, it would be cheaper to use SPLN2, since SPLN2 saves intermediate results from one call to the next.</td>
</tr>
<tr>
<td>SPLNE</td>
<td>Spline evaluation. Evaluates, at a given set of points, a function described by a (previously determined) expansion in terms of B-splines. If many calls with a small number of points of evaluation to SPLNE are anticipated, it would be cheaper to use SPLN2, since SPLN2 saves intermediate results from one call to the next.</td>
</tr>
</tbody>
</table>

**SLATEC**

BSPEV  Calculates the value of the spline and its derivatives at $x$ from the $B$-representation.

BSPPP  Converts the $B$-representation to the piecewise polynomial (PP) form for use with PPVAL.
BSPVD  Calculates the value and all derivatives of order less than \( n_{\text{deriv}} \) of all basis functions which do not vanish at \( x \).
BSPVN  Calculates the value of all (possibly) nonzero basis functions at \( x \).
BVALU  Evaluates the \( B \)-representation of a \( B \)-spline at \( x \) for the function value or any of its derivatives.
DBSPEV  Calculates the value of the spline and its derivatives at \( x \) from the \( B \)-representation. Double precision version.
DBSPPP  Converts the \( B \)-representation to the piecewise polynomial (PP) form for use with PPVAL. Double precision version.
DBSPVD  Calculates the value and all derivatives of order less than \( n_{\text{deriv}} \) of all basis functions which do not vanish at \( x \). Double precision version.
DBSPVN  Calculates the value of all (possibly) nonzero basis functions at \( x \). Double precision version.
DBVALU  Evaluates the \( B \)-representation of a \( B \)-spline at \( x \) for the function value or any of its derivatives. Double precision version.

E8b.  Evaluation of Interpolating or Approximating Functions using Chebyshev Series

**NAG**

E02AEF  Evaluates a polynomial from its Chebyshev-series representation.
E02AHF  Determines the coefficients in the Chebyshev-series representation of the derivative of a polynomial given in Chebyshev-series form.
E02AJF  Determines the coefficients in the Chebyshev-series representation of the indefinite integral of a polynomial given in Chebyshev-series form.
E02AKF  Evaluates a polynomial from its Chebyshev-series representation, allowing an arbitrary index increment for accessing the array of coefficients.
E02CBF  Evaluates a bivariate polynomial from the rectangular array of coefficients in its double Chebyshev series representation.
E8c. Evaluation of Interpolating or Approximating Functions using Cubic Splines

**IMSL**

IBCEVL Evaluation of a bicubic spline.

ICSEVU Evaluation of a cubic spline.

**NAG**

E02BBF Evaluates a cubic spline from its B-spline representation.

E02BCF Evaluates a cubic spline and its first three derivatives from its B-spline representation.

E02DBF Calculates values of a bicubic spline from its B-spline representation.

**PORT**

CSPFE Evaluates a cubic spline function which has already been fit to \( n \) input data pairs \((x, y)\) by PORT program CSPFI. Most users will find it easier to use CSPIN, but the combination CSPFI and CSPFE is potentially more accurate and more efficient.

DCSPFE Double precision version of CSPFE.

**SLATEC**

CHFDV Evaluates a cubic polynomial given in Hermite form and its first derivative at an array of points.

CHFEV Evaluates a cubic polynomial given in Hermite form at an array of points.

PCHFD Evaluates the piecewise cubic interpolant at an array of points and has the capability to evaluate internal derivatives of the interpolant.

PCHFE Evaluates the piecewise cubic interpolant at an array of points.

PCHMC Checks a cubic Hermite function for monotonicity.
Evaluation of Interpolating or Approximating Functions using Splines under Tension

**FITPACK**

**BSURF2**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a B-spline under tension. The subroutine BSURF1 should be called earlier to determine certain necessary parameters. In both BSURF1 and BSURF2, the original grid is assumed to be equally spaced in the \( x \) and \( y \) coordinates.

**BSURF3**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a B-spline under tension. The subroutine BSURF1 should be called earlier to determine certain necessary parameters. In both BSURF1 and BSURF3, the original grid is assumed to be equally spaced in the \( x \) and \( y \) coordinates. The surface values along horizontal lines are output one by one (starting with the line with largest \( y \)-coordinate) by calling a user provided subroutine.

**CURV2**
Interpolates a curve at a given point using a spline under tension. The subroutine CURV1 should be called earlier to determine certain necessary parameters.

**CURVP2**
Interpolates a curve at a given point using a periodic spline under tension. The subroutine CURVP1 should be called earlier to determine certain necessary parameters.

**KURV2**
Performs the mapping of points in the interval \((0,1)\) onto a curve in the plane. The subroutine KURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation both of whose components are splines under tension and functions of the polygonal arclength parameter.

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

**NSURF2**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a B-spline under tension. The subroutine SURF1 or NSURF1 should be called earlier to determine certain necessary parameters.

**NSURF3**
Maps values onto a surface at every point of a grid equally spaced in both \( x \) and \( y \) coordinates. The surface interpolation is performed using a B-spline under tension. The subroutine SURF1 or NSURF1 should be called earlier to determine certain necessary parameters. The surface values along horizontal lines are output one by one (starting with the line with largest \( y \)-coordinate) by calling a user provided subroutine.

**QURV2**
Performs the mapping of points in the interval \((0,1)\) onto a curve in space. The subroutine QURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation all of whose components are splines under tension and functions of the polygonal arclength parameter.
QURVD  Performs the mapping of points in the interval (0,1) onto a curve in space. It also returns the first and second derivatives of the component functions. The subroutine QURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation all of whose components are splines under tension and functions of the polygonal arclength parameter.

QURVP2  Performs the mapping of points in the interval (0,1) onto a closed curve in three-dimensional space. The subroutine QURVP1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation all of whose components are periodic splines under tension and functions of the polygonal arclength parameter.

VAL3BD  Evaluates the function value, the three first partial derivatives and the six second partial derivatives of a tensor product spline under tension in three variables. The subroutine VAL3BD1 should be called earlier to determine certain necessary parameters.

VAL3HD  Evaluates the function value, the three first partial derivatives and the six second partial derivatives of a tensor product spline under tension in three variables.

XSURF2  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine XSURF1 should be called earlier to determine certain necessary parameters. In both XSURF1 and XSURF2, the original grid is assumed to be equally spaced in the $x$ coordinate.

XSURF3  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine XSURF1 should be called earlier to determine certain necessary parameters. In both XSURF1 and XSURF3, the original grid is assumed to be equally spaced in the $x$ coordinate. The surface values along horizontal lines are output one by one (starting with the line with largest $y$-coordinate) by calling a user provided subroutine.

YSURF2  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine YSURF1 should be called earlier to determine certain necessary parameters. In both YSURF1 and YSURF2, the original grid is assumed to be equally spaced in the $y$ coordinate.

YSURF3  Maps values onto a surface at every point of a grid equally spaced in both $x$ and $y$ coordinates. The surface interpolation is performed using a $B$-spline under tension. The subroutine YSURF1 should be called earlier to determine certain necessary parameters. In both YSURF1 and YSURF3, the original grid is assumed to be equally spaced in the $y$ coordinate. The surface values along horizontal lines are output one by one (starting with the line with largest $y$-coordinate) by calling a user provided subroutine.
E9. Mesh Generation for Interpolation or Approximation

NAG

D03MAF Places a triangular mesh over a given 2-dimensional region. The region may have any shape, including one with holes.

PORT

DLUMB Double precision version of LUMB.
DLUMD Double precision version of LUMD.
DMNPB Double precision version of MNPB.
DPUMB Double precision version of PUMB.
DPUMD Double precision version of PUMD.
DUMB Double precision version of UMB.
DUMD Double precision version of UMD.
IDLUMB Double precision version of ILUMB.
IDLUMD Double precision version of ILUMD.
IDMNPB Double precision version of IMNPB.
IDPUMB Double precision version of IPUMB.
IDPUMD Double precision version of IPUMD.
IDUMB Double precision version of IUMB.
IDUMD Double precision version of IUMD.

ILUMB Generates a locally uniform mesh for a B-spline (stack version). Given a basic mesh of points, subdivides each interval into the same number of points. The subdividing points in any given interval are uniformly spaced, but ILUMB generates multiplicities (needed by B-splines) for the two endpoints of the overall interval.

ILUMD Generates locally uniform mesh of distinct points (stack version). Given a basic mesh of points, ILUMD subdivides each interval into the same number of points. The subdividing points in any given interval are uniformly spaced.

IMNPB Generates a B-spline mesh for fitting discrete data (stack version). Creates a B-spline mesh from an array of fitting points, using at least \( n \) fitting points in each mesh interval.

IPUMB Generates a piecewise uniform mesh for a B-spline (stack version). Given a basic mesh of points, each interval in the mesh is subdivided. The number of points of subdivision can vary for each interval. The subdivisions within each interval are uniform, but multiplicities (needed by B-splines) are generated for the two endpoints of the overall interval.

IPUMD Generates a piecewise uniform mesh of distinct points (stack version). Given a basic mesh of points, each interval in the mesh is subdivided. The number of points of subdivision can vary for each interval. The subdividing points in any given interval are uniformly spaced.
IUMB Generates a uniform mesh for a B-spline (stack version). Given the two endpoints of an interval and the number of mesh points desired, generates a uniform mesh. The multiplicities for the two endpoints, needed by B-splines, must also be given.

IUMD Generates a uniform mesh of distinct points (stack version). Given the two endpoints of an interval and the number of mesh points desired, a uniform mesh is generated. Given the two endpoints of an interval and the number of mesh points desired, a uniform mesh is generated.

LUMB Generates a locally uniform mesh for a B-spline. Given a basic mesh of points each interval is subdivided into the same number of points. The subdividing points in any given interval are uniformly spaced, but multiplicities (needed by B-splines) are permitted for the two endpoints of the overall interval.

LUMD Generates a locally uniform mesh of distinct points. Given a basic mesh of points, each interval is subdivided into the same number of points. The subdividing points in any given interval are uniformly spaced.

MNPB Generates a B-spline mesh for fitting discrete data. Creates a B-spline mesh from an array of fitting points, using at least \( N \) fitting points in each mesh interval.

PUMB Generates a piecewise uniform mesh for a B-spline. Given a basic mesh of points, each interval in the mesh is subdivided. The number of points of subdivision can vary for each interval. The subdivisions within each interval are uniform, but multiplicities (needed by B-splines) are generated for the two endpoints of the overall interval.

PUMD Generates piecewise uniform mesh of distinct points. Given a basic mesh of points, each interval in the mesh is subdivided. The number of points of subdivision can vary for each interval. The subdividing points in any given interval are uniformly spaced.

UMB Uniform mesh for a B-spline. Given the two endpoints of an interval and the number of mesh points desired, UMB generates a uniform mesh. The multiplicities for the two endpoints, needed by B-splines, must also be given.

UMD Generates a uniform mesh of distinct points. Given the two endpoints of an interval and the number of mesh points desired, UMD generates a uniform mesh.

**SLATEC**

DINTRV Computes the largest integer \( ileft \) in \( 1 \leq ileft \leq lxt \) such that \( xt(ileft) \leq x \) where \( xt(*) \) is a subdivision of the \( x \) interval. Double precision version.

INTRV Computes the largest integer \( ileft \) in \( 1 \leq ileft \leq lxt \) such that \( xt(ileft) \leq x \) where \( xt(*) \) is a subdivision of the \( x \) interval.
F1a1. Vector Norms and Extremal Elements

**IMSL**

- **DASUM**: Compute double precision sum of absolute values.
- **DNRM2**: Compute the Euclidean length or $L_2$ norm of a double precision vector.
- **ICAMAX**: Find the smallest index of the maximum magnitude of a complex vector.
- **IDAMAX**: Find the smallest index of the maximum magnitude of a double precision vector.
- **ISAMAX**: Find the smallest index of the maximum magnitude of a single precision vector.
- **OCDIS**: Pairwise Euclidean distances between the columns of a matrix.
- **SASUM**: Compute single precision sum of absolute values.
- **SCASUM**: Compute complex sum of absolute values.
- **SCNRM2**: Compute the Euclidean length or $L_2$ norm of a complex vector.
- **SNRM2**: Compute the Euclidean length or $L_2$ norm of a single precision vector.
- **VABMXF**: Maximum absolute value of the elements of a vector or a subset of the elements of a vector.
- **VABMXS**: Maximum absolute value of the elements of a row or column of a matrix stored in symmetric storage mode.
- **VABSXF**: Sum of the absolute values of the elements of a vector or a subset of a vector.
- **VABSMF**: Sum of the absolute values of the elements of a row (or column) of a matrix stored in symmetric storage mode.

**NAG**

- **F05ABF**: Returns the value of the 2-norm of the vector $X$.

**PORT**

- **DNRM2**: Double precision version of SNRM2.
- **SNRM2**: Finds the length (Euclidean norm) of a vector, without underflow or overflow.

**SCILIB**

- **ICAMAX**: Finds the index of the element of a complex vector that has the maximum (absolute real + absolute imaginary). This routine has been optimized for the Cray 1.
- **ISAMAX**: Finds the index of the element of a real vector that has the largest absolute value. This routine has been optimized for the Cray 1.
- **ISAMIN**: Finds the first index of the smallest absolute value element of the vector elements of a real vector. This routine has been optimized for the Cray 1.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISMAX</td>
<td>Finds the first index of the largest element of a vector. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>ISMIN</td>
<td>Finds the first index of the smallest element of a vector. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>SASUM</td>
<td>Sums the absolute values of the elements of a real vector. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>SCASUM</td>
<td>Sums the absolute values of the real and imaginary parts of a complex vector. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>SCNRM2</td>
<td>Computes the Euclidean norm of a complex vector. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>SNRM2</td>
<td>Computes the Euclidean norm of a real vector. This routine has been optimized for the Cray 1.</td>
</tr>
</tbody>
</table>

**SLATEC**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DASUM</td>
<td>Sum of magnitudes of double precision vector components.</td>
</tr>
<tr>
<td>DNRM2</td>
<td>Euclidean length ($L_2$ norm) of double precision vector.</td>
</tr>
<tr>
<td>ICAMAX</td>
<td>Finds largest component of complex vector.</td>
</tr>
<tr>
<td>IDAMAX</td>
<td>Finds largest component of double precision vector.</td>
</tr>
<tr>
<td>ISAMAX</td>
<td>Find largest component of single precision vector.</td>
</tr>
<tr>
<td>SASUM</td>
<td>Sum of magnitudes of single precision vector components.</td>
</tr>
<tr>
<td>SCASUM</td>
<td>Sum of magnitudes of real and imaginary components of complex vector.</td>
</tr>
<tr>
<td>SCNRM2</td>
<td>Unitary norm of complex vector.</td>
</tr>
<tr>
<td>SNRM2</td>
<td>Euclidean length ($L_2$ norm) of single precision vector.</td>
</tr>
</tbody>
</table>

**F1a2. Vector Dot Product**

**IMSL**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDOTC</td>
<td>Compute complex dot product using conjugated vector components.</td>
</tr>
<tr>
<td>CDOTU</td>
<td>Compute complex dot product using unconjugated vector components.</td>
</tr>
<tr>
<td>CZDOTC</td>
<td>Compute complex dot product using conjugated vector components (and double precision accumulation).</td>
</tr>
<tr>
<td>CZDOTU</td>
<td>Compute complex dot product using unconjugated vector components (and double precision accumulation).</td>
</tr>
<tr>
<td>DDOT</td>
<td>Compute double precision dot product.</td>
</tr>
<tr>
<td>DSDOT</td>
<td>Compute single precision dot product using double precision accumulation.</td>
</tr>
</tbody>
</table>
**SDOT**  Compute single precision dot product.

**SDSDOT**  Compute single precision dot product and add a constant using double precision accumulation.

**VIPRFF**  Vector inner product of two vectors or subsets of two vectors.

**VIPRSS**  Vector inner product of two vectors each of which is part of some matrix stored in symmetric mode.

**NAG**

**F01DAF**  Returns the sum of an initial value and a scalar product, using basic precision arithmetic.

**F01DBF**  Returns the sum of an initial value and a scalar product, using additional precision arithmetic.

**F01DCF**  Computes the value of a complex scalar product and subtracts it from a complex initial value, using basic precision arithmetic.

**F01DDF**  Computes the value of a complex scalar product and subtracts it from a complex initial value, using additional precision arithmetic.

**F01DEF**  Returns the value of the scalar product of two arrays of length $N$, using basic precision arithmetic.

**X03AAF**  Calculates the value of a scalar product using basic or additional precision and adds it to a basic or additional precision initial value.

**X03ABF**  Calculates the value of a complex scalar product using basic or additional precision and adds it to a complex initial value.

**SCILIB**

**CDOTC**  Cray 1 optimized routine for computing the inner product of a complex vector with the conjugate of a complex vector.

**CDOTU**  Cray 1 optimized routine for computing the inner product of two complex vectors.

**SDOT**  Computes the inner product of two real vectors. This routine has been optimized for the Cray 1.

**SPDOT**  Computes the sparse inner product of two real vectors. This routine has been optimized for the Cray 1.

**SLATEC**

**CDCDOT**  Complex inner product with result accumulated in double precision.

**CDOTC**  Computes the dot product of complex vectors, uses complex conjugate of first vector.

**CDOTU**  Inner product of complex vectors.

**DCDOT**  Computes the dot product of 2 complex vectors, $x$ and $y$, e.g. $CX \text{ DOT } CY$, or, $CX$ conjugate $\text{ DOT } CY$. The real and imaginary parts of $x$ and $y$ are converted to double precision, the dot product accumulation is done in double precision and the output is given as 2 double precision numbers, corresponding to the real and imaginary part of the result.
### NCAR Software Catalog

**DDOT**
Double precision inner product of double precision vectors.

**DQDOTA**
Double precision inner product with extended precision accumulation and result.

**DQDOTI**
Double precision inner product with extended precision accumulation and result.

**DSDOT**
Double precision inner product of single precision vectors.

**SDOT**
Single precision inner product of single precision vectors.

**SDSDOT**
Single precision result with inner product accumulated in double precision.

### F1a3. Vector Copy or Swap

#### IMSL

**CCOPY**
Copy a vector \( x \) to a vector \( y \), both complex.

**CSWAP**
Interchange vectors \( x \) and \( y \), both complex.

**DCOPY**
Copy a vector \( x \) to a vector \( y \), both double precision.

**DSWAP**
Interchange vectors \( x \) and \( y \), both double precision.

**SCOPY**
Copy a vector \( x \) to a vector \( y \), both single precision.

**SSWAP**
Interchange vectors \( x \) and \( y \), both single precision.

#### NAG

**F01CNF**
Copies a vector of length \( M \) into a row of a matrix.

**F01CPF**
Copies the contents of a vector into a second vector.

**F01CQF**
Sets the elements of a vector to zero.

#### PORT

**MOVEBC**
Moves a specified number of values from one complex array to another using a backward loop (which has a different effect than a forward loop if the arrays overlap).

**MOVEBD**
Moves a specified number of values from one double precision array to another using a backward loop (which has a different effect than a forward loop if the arrays overlap).

**MOVEBI**
Moves a specified number of values from one integer array to another using a backward loop (which has a different effect than a forward loop if the arrays overlap).

**MOVEBL**
Moves a specified number of values from one logical array to another using a backward loop (which has a different effect than a forward loop if the arrays overlap).
MOVEBR  Moves a specified number of values from one real array to another using a backward loop (which has a different effect than a forward loop if the arrays overlap).

MOVEFC  Moves a specified number of values from one complex array to another using a forward loop.

MOVEFD  Moves a specified number of values from one double precision array to another using a forward loop.

MOVEFI  Moves a specified number of values from one integer array to another using a forward loop.

MOVEFL  Moves a specified number of values from one logical array to another using a forward loop.

MOVEFR  Moves a specified number of values from one real array to another using a forward loop.

SETC    Sets a specified number of values of a complex array equal to a constant.

SETD    Sets a specified number of values of a double precision array equal to a constant.

SETI    Sets a specified number of values of an integer array equal to a constant.

SETL    Sets a specified number of values of a logical array equal to a constant.

SETR    Sets a specified number of values of a real array equal to a constant.

SCILIB

CCOPY   Copies a complex vector into another complex vector. This routine has been optimized for the Cray 1.

CSWAP   Exchanges two complex vectors. This routine has been optimized for the Cray 1.

SCOPY   Copies a real vector into another real vector. This routine has been optimized for the Cray 1.

SSWAP   Exchanges two real vectors. This routine has been optimized for the Cray 1.

SLATEC

CCOPY   Complex vector copy \( y = z \).

CSWAP   Interchange complex vectors.

DCOPY   Double precision vector copy \( y = z \).

DSWAP   Interchange double precision vectors.

SCOPY   Copy single precision vector \( y = z \).

SCOPYM  Copy negative of real SX to real SY.

SSWAP   Interchanges single precision vectors.
NCAR Software Catalog

Fla5. Vector Scaling (v1 = s*v1 + v2)

**IMSL**

CAXPY  Compute a constant times a vector plus a vector, all complex.
CSCAL  Compute a complex constant times a complex vector.
CSSCAL Compute a real constant times a complex vector.
DAXPY  Compute a constant times a vector plus a vector, all double precision.
DSCAL  Compute a double precision constant times a double precision vector.
SAXPY  Compute a constant times a vector plus a vector, all single precision.
SSCAL  Compute a single precision constant times a single precision vector.

**SCILIB**

CAXPY  Adds a complex multiple of one complex vector to another. This routine has been optimized for the Cray 1.
CSCAL  Scales a complex vector by a given complex number. This routine has been optimized for the Cray 1.
CSSCAL Scales a complex vector by a given real number. This routine has been optimized for the Cray 1.
SAXPY  Adds a scalar multiple of one real vector to another. This routine has been optimized for the Cray 1.
SPAXPY Adds a scalar multiple of one sparse real vector to another. This routine has been optimized for the Cray 1.
SSCAL  Scales a real vector by a given real number. This routine has been optimized for the Cray 1.

**SLATEC**

CAXPY  Complex computation y = ax + y.
CSCAL  Complex vector scale z = az.
CSSCAL Scale a complex vector.
DAXPY  Computes y = ax + y. Double precision version.
DSCAL  Double precision vector scale z = az.
SAXPY  Single precision computation y = ax + y.
SSCAL  Single precision vector scale z = az.
F1a6. Elementary Rotations (Givens Transformations)

**IMSL**
- **DROT**: Apply Givens plane rotation (double precision).
- **DROTG**: Construct Givens plane rotation (double precision).
- **DROTM**: Apply a modified Givens plane rotation (double precision).
- **DROTMG**: Construct a modified Givens plane rotation (double precision).
- **SROT**: Apply Givens plane rotation (single precision).
- **SROTG**: Construct Givens plane rotation (single precision).
- **SROTM**: Apply a modified Givens plane rotation (single precision).
- **SROTMG**: Construct a modified Givens plane rotation (single precision).

**SCILIB**
- **CROT**: Applies the complex Givens rotation computed by the subroutine CROTG. This routine has been optimized for the Cray 1.
- **CROTG**: Computes the elements of a complex $2 \times 2$ Givens rotation matrix that zeros out the second element of a given complex 2-vector. This routine has been optimized for the Cray 1.
- **SROT**: Applies the Givens rotation computed by the subroutine SROTG. This routine has been optimized for the Cray 1.
- **SROTG**: Computes the elements of a $2 \times 2$ Givens rotation matrix that zeros out the second element of a given 2-vector. This routine has been optimized for the Cray 1.
- **SROTM**: Applies the modified Givens rotation computed by the subroutine SROTMG. This routine has been optimized for the Cray 1.
- **SROTMG**: Computes the elements of a modified Givens rotation matrix. This routine has been optimized for the Cray 1.

**SLATEC**
- **CROTG**: Construct a complex Givens transformation.
- **CSROT**: Applies a plane rotation to complex vectors.
- **DROT**: Applies double precision Givens rotation.
- **DROTG**: Constructs double precision plane Givens rotation.
- **DROTM**: Applies double precision modified Givens transformation.
- **DROTMG**: Constructs double precision modified Givens transformation.
- **SROT**: Apply single precision Givens rotation.
- **SROTG**: Construct single precision plane Givens rotation.
SROTM | Applies single precision modified Givens transformation.
SROTMG | Constructs single precision modified Givens transformation.

F1a7. Elementary Reflections (Householder Transformations)

IMSL

VHS12 | Real Householder transformation – computation and applications.
VHSH2C | Complex Householder transformation to zero a single element of a matrix.
VHSH2R | Real Householder transformation to zero a single element of a matrix.
VHSH3R | Real Householder transformation to zero two elements of a matrix.

F1a8. Other Elementary Vector Operations, Including Vectorization Aids

ITPACK

PERVEC | Permutes a real vector as dictated by the permutation vector, p. If p(i) = j, then v(j) gets v(i).
VFILL | Fills a vector, v, with a constant value, val.

LINPACK

BLAS | A collection of linear algebra subprograms which perform basic vector operations (e.g. vector norm, dot product etc.).
BLASCRAY | A collection of linear algebra subprograms which perform basic vector operations (e.g. vector norm, dot product etc.) not available in $SCILIB.$

SCILIB

FOLR | Cray 1 optimized routine for solving the first-order linear recurrence $c_i = b_i$, $c_i = -a_i c_{i-1} + b_i$, $i = 2, 3, \ldots, n$. The solution vector overwrites the input vector.
FOLR2 | Cray 1 optimized routine for solving the first-order linear recurrence $c_i = b_i$, $c_i = -a_i c_{i-1} + b_i$, $i = 2, 3, \ldots, n$. The solution vector does not overwrite the input vector.
<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLR2P</td>
<td>Cray 1 optimized routine for solving the first-order linear recurrence $c_i = b_i$, $c_i = a_i c_{i-1} + b_i$, $i = 2, 3, \ldots, n$. The solution vector does not overwrite the input vector.</td>
</tr>
<tr>
<td>FOLRN</td>
<td>Cray 1 optimized routine solving for the last term of the first-order linear recurrence $r_i = b_i$, $r_i = -a_i r_{i-1} + b_i$, $i = 2, 3, \ldots, n$. This routine allows for efficient evaluation of polynomials using Horner's method.</td>
</tr>
<tr>
<td>FOLRP</td>
<td>Cray 1 optimized routine for solving the first-order linear recurrence $c_i = b_i$, $c_i = a_i c_{i-1} + b_i$, $i = 2, 3, \ldots, n$. The solution vector overwrites the input vector.</td>
</tr>
<tr>
<td>GATHER</td>
<td>Gathers a vector from a source vector, using a vector of indexes to determine which elements are accessed. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>ILLZ</td>
<td>Returns the number of positive or zero values (false values) before the first negative value (true value) of a vector declared INTEGER or REAL (LOGICAL). This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>ILSUM</td>
<td>Returns the total number of negative values (true values) in a vector declared INTEGER or REAL (LOGICAL). This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>SCATTER</td>
<td>Scatters a vector into another vector, using a vector of indexes to determine which elements are changed. This routine has been optimized for the Cray 1.</td>
</tr>
<tr>
<td>SOLR</td>
<td>Cray 1 optimized routine for solving the second-order linear recurrence $c_{i+2} = a_i c_{i+1} + b_i c_i$, $i = 1, 2, \ldots, n$.</td>
</tr>
<tr>
<td>SOLR3</td>
<td>Cray 1 optimized routine for solving the second-order linear recurrence of three terms $c_i = c_i + a_{i-2} c_{i-1} + b_{i-2} c_{i-2}$, $i = 3, \ldots, n$.</td>
</tr>
<tr>
<td>SOLRN</td>
<td>Cray 1 optimized routine solving for the last term of the second-order linear recurrence $c_{i+2} = a_i c_{i+1} + b_i c_i$, $i = 1, 2, \ldots, n$.</td>
</tr>
<tr>
<td>SSUM</td>
<td>Sums the elements of a real vector. This routine has been optimized for the Cray 1.</td>
</tr>
</tbody>
</table>

### F1b3. Matrix Multiplication

**IMSL**

- **VMULBB** Matrix multiplication (band storage mode).
- **VMULBF** Matrix multiplication (band by full matrices).
- **VMULBS** Matrix multiplication (band by symmetric).
- **VMULFB** Matrix multiplication (full by band matrices).
- **VMULFF** Matrix multiplication (full storage mode).
- **VMULFM** Matrix multiplication of the transpose of matrix $A$ by matrix $B$ (full storage mode).
- **VMULFP** Matrix multiplication of matrix $A$ by the transpose of matrix $B$ (full storage mode).

Linear Algebra 108
VMULFQ  Matrix multiplication (full by band symmetric matrices).
VMULFS  Matrix multiplication (full by symmetric matrices).
VMULQB  Matrix multiplication (band symmetric by band matrices).
VMULQF  Matrix multiplication (band symmetric by full matrices).
VMULQQ  Matrix multiplication (band symmetric storage mode).
VMULQS  Matrix multiplication (band symmetric by symmetric matrices).
VMULSB  Matrix multiplication (symmetric by band matrices).
VMULSF  Matrix multiplication (symmetric by full matrices).
VMULSQ  Matrix multiplication (symmetric by band symmetric matrices).
VMULSS  Matrix multiplication (symmetric storage mode).
VTPROF  Transpose product of matrix (full storage mode).
VTPROS  Transpose product of a matrix (symmetric storage mode).

**NAG**

F01CKF  Returns with the result of the multiplication of two matrices $B$ and $C$ in the
matrix $A$, with the option to overwrite $B$ or $C$.
F01CLF  Post-multiplies the matrix $B$ with the transpose of the matrix $C$ and places the
result in the matrix $A$.
F01CSF  Forms the product $c = Ab$ where $b$ is a vector and $A$ is a symmetric matrix
whose lower triangle is stored by rows in a one-dimensional array.

**SCILIB**

MXM  Computes the product of two matrices, assuming the elements of each matrix are
contiguous in memory. This routine has been optimized for the Cray 1.
MXMA  Computes the product of two matrices, allowing for arbitrary spacing of matrix
elements. This routine has been optimized for the Cray 1.

---

**F1b7. Matrix Storage Mode Conversion**

**IMSL**

VCVTBF  Storage mode conversion of matrices (band to full storage mode).
VCVTCH  Storage mode conversion of matrices (full complex to Hermitian).
VCVTFB  Storage mode conversion of matrices (full to band storage mode).
VCVTFQ  Storage mode conversion (full to band symmetric storage mode).
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCVTFS</td>
<td>Storage mode conversion of matrices (full to symmetric).</td>
</tr>
<tr>
<td>VCVTHC</td>
<td>Storage mode conversion of matrices (Hermitian to full complex).</td>
</tr>
<tr>
<td>VCVTQF</td>
<td>Storage mode conversion (band symmetric to full storage mode).</td>
</tr>
<tr>
<td>VCVTQS</td>
<td>Storage mode conversion (band symmetric to symmetric storage mode).</td>
</tr>
<tr>
<td>VCVTSF</td>
<td>Storage mode conversion of matrices (symmetric to full).</td>
</tr>
<tr>
<td>VCVTSQ</td>
<td>Storage mode conversion (symmetric to band symmetric storage mode).</td>
</tr>
</tbody>
</table>

F1b8. Other Elementary Matrix Operations

**IMSL**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDTRGI</td>
<td>Transeneration of the columns of a matrix (in-core version).</td>
</tr>
<tr>
<td>BDTRGO</td>
<td>Transeneration of the columns of a matrix (out-of-core version).</td>
</tr>
<tr>
<td>VNRMF1</td>
<td>1-norm of matrices (full storage mode).</td>
</tr>
<tr>
<td>VNRMF2</td>
<td>Euclidean-norm of matrices (full storage mode).</td>
</tr>
<tr>
<td>VNRMFI</td>
<td>Infinity-norm matrices (full storage mode).</td>
</tr>
<tr>
<td>VNRMS1</td>
<td>1-norm of matrices (symmetric storage mode).</td>
</tr>
<tr>
<td>VNRMS2</td>
<td>Euclidean-norm of matrices (symmetric storage mode).</td>
</tr>
<tr>
<td>VPOLYF</td>
<td>Matrix polynomial (full storage mode).</td>
</tr>
<tr>
<td>VTRAN</td>
<td>Transpose a rectangular matrix.</td>
</tr>
<tr>
<td>VUABQ</td>
<td>Matrix addition (band + band symmetric matrices).</td>
</tr>
<tr>
<td>VUAFB</td>
<td>Matrix addition (full + band matrices).</td>
</tr>
<tr>
<td>VUAFQ</td>
<td>Matrix addition (full + band symmetric matrices).</td>
</tr>
<tr>
<td>VUAFS</td>
<td>Matrix addition (full + symmetric matrices).</td>
</tr>
<tr>
<td>VUASB</td>
<td>Matrix addition (symmetric + band matrices).</td>
</tr>
<tr>
<td>VUASQ</td>
<td>Matrix addition (symmetric + band symmetric matrices).</td>
</tr>
</tbody>
</table>

**NAG**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01CAF</td>
<td>Sets elements of an $m \times n$ matrix $A$ to zero.</td>
</tr>
<tr>
<td>F01CBF</td>
<td>Sets the elements $A_{i,j}$ to one if $i=j$ and zero otherwise, where $1 \leq i \leq m$ and $1 \leq j \leq n$.</td>
</tr>
<tr>
<td>F01CDF</td>
<td>Adds elements of the $m \times n$ matrices $B$ and $C$ and stores the results in elements of the matrix $A$.</td>
</tr>
<tr>
<td>F01CEF</td>
<td>Subtracts elements of the matrix $C$ from elements of the matrix $B$ and stores the results in elements of the matrix $A$.</td>
</tr>
</tbody>
</table>

Linear Algebra 110
FO1CFF: Copies elements of the matrix B into different positions in the matrix A.
FO1CGF: Adds elements of the matrix B to elements in different positions in the matrix A.
FO1CHF: Subtracts elements of the matrix B from elements in a different position in the matrix A.
FO1CMF: Copies elements of one matrix into a second matrix.
FO1CRF: Re-orders the elements of a vector of length $m \times n$, containing an $m \times n$ matrix, A, so that the new vector contains the transpose matrix.

SCILIB

MXV: Computes the product of matrix times vector with unit spacing between row elements. This routine has been optimized for the Cray 1.
MXVA: Computes the product of a matrix times vector, allowing for arbitrary spacing of matrix elements. This routine has been optimized for the Cray 1.

F2a1. Eigenvalue Problems: Reduction to Symmetric Tridiagonal Form

EISPACK

BANDR: Reduces a real symmetric band matrix to a symmetric tridiagonal matrix using and optionally accumulating orthogonal similarity transformations.
FIGI: Given a nonsymmetric tridiagonal matrix such that the products of corresponding pairs of off-diagonal elements are all non-negative, this subroutine reduces it to a symmetric tridiagonal matrix with the same eigenvalues. If, further, a zero product only occurs when both factors are zero, the reduced matrix is similar to the original matrix.
FIGI2: Given a nonsymmetric tridiagonal matrix such that the products of corresponding pairs of off-diagonal elements are all non-negative, and zero only when both factors are zero, this subroutine reduces it to a symmetric tridiagonal matrix using and accumulating diagonal similarity transformations.
HTRID3: Reduces a complex Hermitian matrix, stored as a single square array, to a real symmetric tridiagonal matrix using unitary similarity transformations.
HTRIDI: Reduces a complex Hermitian matrix to a real symmetric tridiagonal matrix using unitary similarity transformations.
TRED1: Reduces a real symmetric matrix to a symmetric tridiagonal matrix using orthogonal similarity transformations.
TRED2: Reduces a real symmetric matrix to a symmetric tridiagonal matrix using and accumulating orthogonal similarity transformations.
TRED3: Reduces a real symmetric matrix, stored as a one-dimensional array, to a symmetric tridiagonal matrix using orthogonal similarity transformations.
**EISPKD**

**BANDR**  
Reduces a real double precision symmetric band matrix to a double precision symmetric tridiagonal matrix using and optionally accumulating orthogonal similarity transformations.

**FIGI**  
Given a double precision nonsymmetric tridiagonal matrix such that the products of corresponding pairs of off-diagonal elements are all non-negative, this subroutine reduces it to a double precision symmetric tridiagonal matrix with the same eigenvalues. If, further, a zero product only occurs when both factors are zero, the reduced matrix is similar to the original matrix.

**FIGI2**  
Given a double precision nonsymmetric tridiagonal matrix such that the products of corresponding pairs of off-diagonal elements are all non-negative, and zero only when both factors are zero, this subroutine reduces it to a double precision symmetric tridiagonal matrix using and accumulating diagonal similarity transformations.

**HTRID3**  
Reduces a double precision complex Hermitian matrix, stored as a single square array, to a double precision real symmetric tridiagonal matrix using unitary similarity transformations.

**HTRIDI**  
Reduces a double precision complex Hermitian matrix to a double precision real symmetric tridiagonal matrix using unitary similarity transformations.

**TRED1**  
Reduces a double precision real symmetric matrix to a double precision symmetric tridiagonal matrix using orthogonal similarity transformations.

**TRED2**  
Reduces a double precision real symmetric matrix to a double precision symmetric tridiagonal matrix using and accumulating orthogonal similarity transformations.

**TRED3**  
Reduces a double precision real symmetric matrix, stored as a one-dimensional array, to a double precision symmetric tridiagonal matrix using orthogonal similarity transformations.

**IMSL**

**EHOUSH**  
Reduction of a complex Hermitian matrix to real symmetric tridiagonal form.

**EHOUSS**  
Reduction of a symmetric matrix to symmetric tridiagonal form using a Householder reduction.

**NAG**

**F01AGF**  
Gives the Householder reduction of a real symmetric matrix to tridiagonal form for use in F02BEF, F02AVF and F02BFF.

**F01AJF**  
Gives the Householder reduction of a real symmetric matrix $A$ to tridiagonal form for use in F02AMF.

**F01AYF**  
Gives the Householder reduction of a real symmetric matrix $A$ to tridiagonal form for use in F02BEF, F02AVF and F02BFF. The routine is similar to F01AGF but is more economical in storage.

**F01BCF**  
Gives the Householder reduction of a complex Hermitian matrix to tridiagonal form for use in F02AVF or F02AYF.

**F01BWF**  
Reduces a symmetric band matrix to tridiagonal form. This routine will normally be used in conjunction with F02AVF to find all the eigenvalues of $A$. For
selected eigenvalues there is a choice between F02BMF and the combination of FO1BWF and F02BFF.

### SCILIB

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANDR</td>
<td>Cray 1 optimized version of the EISPACK routine; reduces a real symmetric band matrix to a symmetric tridiagonal matrix using and optionally accumulating orthogonal similarity transformations.</td>
</tr>
<tr>
<td>FIGI</td>
<td>Cray 1 optimized version of the EISPACK routine; given a nonsymmetric tridiagonal matrix such that the products of corresponding pairs of off-diagonal elements are all non-negative, this subroutine reduces it to a symmetric tridiagonal matrix with the same eigenvalues. If, further, a zero product only occurs when both factors are zero, the reduced matrix is similar to the original matrix.</td>
</tr>
<tr>
<td>FIGI2</td>
<td>Cray 1 optimized version of the EISPACK routine; given a nonsymmetric tridiagonal matrix such that the products of corresponding pairs of off-diagonal elements are all non-negative, and zero only when both factors are zero, this subroutine reduces it to a symmetric tridiagonal matrix using and accumulating diagonal similarity transformations.</td>
</tr>
<tr>
<td>HTRID3</td>
<td>Cray 1 optimized version of the EISPACK routine that reduces a complex Hermitian matrix, stored as a single square array, to a real symmetric tridiagonal matrix using unitary similarity transformations.</td>
</tr>
<tr>
<td>HTRIDI</td>
<td>Cray 1 optimized version of the EISPACK routine that reduces a complex Hermitian matrix to a real symmetric tridiagonal matrix using unitary similarity transformations.</td>
</tr>
<tr>
<td>TRED1</td>
<td>Cray 1 optimized version of the EISPACK routine; reduces a real symmetric matrix to a symmetric tridiagonal matrix using orthogonal similarity transformations.</td>
</tr>
<tr>
<td>TRED2</td>
<td>Cray 1 optimized version of the EISPACK routine; reduces a real symmetric matrix to a symmetric tridiagonal matrix using and accumulating orthogonal similarity transformations.</td>
</tr>
<tr>
<td>TRED3</td>
<td>Cray 1 optimized version of the EISPACK routine; reduces a real symmetric matrix, stored as a one-dimensional array, to a symmetric tridiagonal matrix using orthogonal similarity transformations.</td>
</tr>
</tbody>
</table>

### SLATEC

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANDR</td>
<td>Reduces real symmetric band matrix to symmetric tridiagonal matrix and, optionally, accumulates orthogonal similarity transformations.</td>
</tr>
<tr>
<td>FIGI</td>
<td>Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.</td>
</tr>
<tr>
<td>FIGI2</td>
<td>Transforms certain real non-symmetric tridiagonal matrix to symmetric tridiagonal matrix.</td>
</tr>
<tr>
<td>HTRID3</td>
<td>Reduces complex Hermitian (packed) matrix to real symmetric tridiagonal matrix by unitary similarity transformations.</td>
</tr>
<tr>
<td>HTRIDI</td>
<td>Reduces complex Hermitian matrix to real symmetric tridiagonal matrix using unitary similarity transformations.</td>
</tr>
<tr>
<td>TRED1</td>
<td>Reduce real symmetric matrix to symmetric tridiagonal matrix using orthogonal similarity transformations.</td>
</tr>
</tbody>
</table>
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.

F2a2.  Eigenvalue Problems: Reduction to Hessenberg Form

EISPACK

COMHES  Given a complex general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by stabilized elementary similarity transformations.

CORTH  Given a complex general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by unitary similarity transformations.

ELMHES  Given a real general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by stabilized elementary similarity transformations.

ELTRAN  Accumulates the stabilized elementary similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ELMHES.

ORTHES  Given a real general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by orthogonal similarity transformations.

ORTRAN  Accumulates the orthogonal similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ORTHES.

EISPKD

COMHES  Given a double precision complex general matrix, this subroutine reduces a submatrix situated in rows and columns low through high to double precision upper Hessenberg form by stabilized elementary similarity transformations.

CORTH  Given a double precision complex general matrix, this subroutine reduces a submatrix situated in rows and columns low through high to upper Hessenberg form by unitary similarity transformations.

ELMHES  Given a double precision real general matrix, this subroutine reduces a submatrix situated in rows and columns low through high to upper Hessenberg form by stabilized elementary similarity transformations.

ELTRAN  Accumulates the stabilized elementary similarity transformations used in the reduction of a double precision real general matrix to upper Hessenberg form by ELMHES.

ORTHES  Given a double precision real general matrix, this subroutine reduces a submatrix situated in rows and columns low through high to upper Hessenberg form by orthogonal similarity transformations.
Accumulates the orthogonal similarity transformations used in the reduction of a double precision real general matrix to upper Hessenberg form by ORTHES.

**IMSL**

**EHESSC** Reduction of a general complex matrix to complex upper Hessenberg form.

**EHESSF** Reduction of a nonsymmetric matrix to upper Hessenberg form by orthogonal transformations.

**ELZHC** Simultaneously reduce two complex matrices $A$ and $B$, $A$ to upper Hessenberg and $B$ to upper triangular form.

**EQZQF** Hessenberg reduction for the generalized eigenvalue problem $Ax = \lambda Bx$. Reduction of $A$ to upper Hessenberg form and $B$ to upper triangular form.

**NAG**

**F01AKF** Reduces a real unsymmetric matrix to upper Hessenberg form.

**F01AMF** Reduces a complex unsymmetric matrix to complex upper Hessenberg form.

**F01APF** Forms the matrix of accumulated transformations from information left by F01AKF.

**SCILIB**

**COMHES** Cray 1 optimized version of the EISPACK routine; given a complex general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by stabilized elementary similarity transformations.

**CORTH** Cray 1 optimized version of the EISPACK routine; given a complex general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by unitary similarity transformations.

**ELMHES** Cray 1 optimized version of the EISPACK routine; given a real general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by stabilized elementary similarity transformations.

**ELTRAN** Cray 1 optimized version of the EISPACK routine; accumulates the stabilized elementary similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ELMHES.

**ORTHES** Cray 1 optimized version of the EISPACK routine; given a real general matrix, this subroutine reduces a submatrix situated in rows and columns low through igh to upper Hessenberg form by orthogonal similarity transformations.

**ORTRAN** Cray 1 optimized version of the EISPACK routine; accumulates the orthogonal similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ORTHES.
**SLATEC**

<table>
<thead>
<tr>
<th>COMHES</th>
<th>Reduces complex general matrix to complex upper Hessenberg form using stabilized elementary similarity transformations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORTH</td>
<td>Reduces complex general matrix to complex upper Hessenberg using unitary similarity transformations.</td>
</tr>
<tr>
<td>ELMHES</td>
<td>Reduces real general matrix to upper Hessenberg form stabilized elementary similarity transformations.</td>
</tr>
<tr>
<td>ELTRAN</td>
<td>Accumulates the stabilized elementary similarity transformations used in the reduction of a real general</td>
</tr>
<tr>
<td></td>
<td>matrix to upper Hessenberg form by ELMHES.</td>
</tr>
<tr>
<td>ORTHES</td>
<td>Reduces real general matrix to upper Hessenberg form orthogonal similarity transformations.</td>
</tr>
<tr>
<td>ORTRAN</td>
<td>Accumulates orthogonal similarity transformations in reduction of real general matrix by ORTHES.</td>
</tr>
</tbody>
</table>

**F2c1. Eigenvalues and Eigenvectors of Tridiagonal Matrices**

**EISPACK**

<p>| BAKVEC       | Forms the eigenvectors of a nonsymmetric tridiagonal matrix by back transforming those of the corresponding |
|              | symmetric matrix determined by FIGI.                                                                     |
| BISECT       | Finds those eigenvalues of a tridiagonal symmetric matrix which lie in a specified interval, using bisection. |
| IMTQL1       | Finds the eigenvalues of a symmetric tridiagonal matrix by the implicit QL method.                        |
| IMTQL2       | Finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the implicit QL method.       |
|              | The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full  |
|              | matrix to tridiagonal form.                                                                             |
| IMTQLV       | Finds the eigenvalues of a symmetric tridiagonal matrix by the implicit QL method and associates with    |
|              | them their corresponding submatrix indices.                                                             |
| RATQR        | Finds the algebraically smallest or largest eigenvalues of a symmetric tridiagonal matrix by the rational |
|              | QR method with Newton corrections.                                                                     |
| RST          | Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find  |
|              | the eigenvalues and eigenvectors (if desired) of a real symmetric tridiagonal matrix.                    |
| RT           | Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find  |
|              | the eigenvalues and eigenvectors (if desired) of a special real tridiagonal matrix.                      |
| TINVIT       | Finds those eigenvectors of a tridiagonal symmetric matrix corresponding to specified eigenvalues, using |
|              | inverse iteration.                                                                                      |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQL1</td>
<td>Finds the eigenvalues of a symmetric tridiagonal matrix by the QL method.</td>
</tr>
<tr>
<td>TQL2</td>
<td>Finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.</td>
</tr>
<tr>
<td>TQLRAT</td>
<td>Finds the eigenvalues of a symmetric tridiagonal matrix by the rational QL method.</td>
</tr>
<tr>
<td>TRIDIB</td>
<td>Finds those eigenvalues of a tridiagonal symmetric matrix between specified boundary indices, using bisection.</td>
</tr>
<tr>
<td>TSTURM</td>
<td>Finds those eigenvalues of a tridiagonal symmetric matrix which lie in a specified interval and their associated eigenvectors, using bisection and inverse iteration.</td>
</tr>
</tbody>
</table>

**EISPKD**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAKVEC</td>
<td>Forms the eigenvectors of a double precision nonsymmetric tridiagonal matrix by back transforming those of the corresponding double precision symmetric matrix determined by FIGI.</td>
</tr>
<tr>
<td>BISECT</td>
<td>Finds those eigenvalues of a tridiagonal double precision symmetric matrix which lie in a specified interval, using bisection.</td>
</tr>
<tr>
<td>IMTQL1</td>
<td>Finds the eigenvalues of a double precision symmetric tridiagonal matrix by the implicit QL method.</td>
</tr>
<tr>
<td>IMTQL2</td>
<td>Finds the eigenvalues and eigenvectors of a double precision symmetric tridiagonal matrix by the implicit QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.</td>
</tr>
<tr>
<td>IMTQLV</td>
<td>Finds the eigenvalues of a double precision symmetric tridiagonal matrix by the implicit QL method and associates with them their corresponding submatrix indices.</td>
</tr>
<tr>
<td>RATQR</td>
<td>Finds the algebraically smallest or largest eigenvalues of a double precision symmetric tridiagonal matrix by the rational QR method with Newton corrections.</td>
</tr>
<tr>
<td>RST</td>
<td>Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision real symmetric tridiagonal matrix.</td>
</tr>
<tr>
<td>RT</td>
<td>Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a special double precision real tridiagonal matrix.</td>
</tr>
<tr>
<td>TINVIT</td>
<td>Finds those eigenvectors of a double precision symmetric tridiagonal matrix corresponding to specified eigenvalues, using inverse iteration.</td>
</tr>
<tr>
<td>TQL1</td>
<td>Finds the eigenvalues of a double precision symmetric tridiagonal matrix by the QL method.</td>
</tr>
<tr>
<td>TQL2</td>
<td>Finds the eigenvalues and eigenvectors of a double precision symmetric tridiagonal matrix by the QL method. The eigenvectors of a full double precision symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.</td>
</tr>
<tr>
<td>TQLRAT</td>
<td>Finds the eigenvalues of a double precision symmetric tridiagonal matrix by the rational QL method.</td>
</tr>
</tbody>
</table>
TRIDIB  Finds those eigenvalues of a tridiagonal double precision symmetric matrix between specified boundary indices, using bisection.

TSTURM  Finds those eigenvalues of a tridiagonal double precision symmetric matrix which lie in a specified interval and their associated eigenvectors, using bisection and inverse iteration.

**IMSL**

EHBCKH  Back transformation of the eigenvectors of a real symmetric tridiagonal matrix obtained from the Householder reduction of a Hermitian matrix.

EQRT1S  Smallest or largest $m$ eigenvalues of a symmetric tridiagonal matrix.

EQRT2S  Eigenvalues and (optionally) eigenvectors of a symmetric tridiagonal matrix using the QL method.

EQRT3S  The smallest (or largest) eigenvalues of a tridiagonal matrix in algebraic value whose sum exceeds a given value.

**LOCLIB**

EIGSTM  Computes the algebraically smallest or largest $m$ eigenvalues and corresponding eigenvectors of a real symmetric tridiagonal matrix.

EIGSTS  Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric tridiagonal matrix.

**NAG**

F01AHF  Derives eigenvectors of a real symmetric matrix from eigenvectors of the tridiagonal form where the tridiagonal matrix was produced by F01AGF.

**SCILIB**

BAKVEC  Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a nonsymmetric tridiagonal matrix by back transforming those of the corresponding symmetric matrix determined by FIGI.

BISECT  Cray 1 optimized version of the EISPACK routine; finds those eigenvalues of a tridiagonal symmetric matrix which lie in a specified interval, using bisection.

IMTQL1  Cray 1 optimized version of the EISPACK routine; finds the eigenvalues of a symmetric tridiagonal matrix by the implicit QL method.

IMTQL2  Cray 1 optimized version of the EISPACK routine; finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the implicit QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.

IMTQLV  Cray 1 optimized version of the EISPACK routine; finds the eigenvalues of a symmetric tridiagonal matrix by the implicit QL method and associates with them their corresponding submatrix indices.

RATQR  Cray 1 optimized version of the EISPACK routine; finds the algebraically smallest or largest eigenvalues of a symmetric tridiagonal matrix by the rational QR method with Newton corrections.
### TINVIT
Cray 1 optimized version of the EISPACK routine; finds those eigenvectors of a tridiagonal symmetric matrix corresponding to specified eigenvalues, using inverse iteration.

### TQL1
Cray 1 optimized version of the EISPACK routine; finds the eigenvalues of a symmetric tridiagonal matrix by the QL method.

### TQL2
Cray 1 optimized version of the EISPACK routine; finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.

### TQLRAT
Cray 1 optimized version of the EISPACK routine; finds the eigenvalues of a symmetric tridiagonal matrix by the rational QL method.

### TRIDIB
Cray 1 optimized version of the EISPACK routine; finds those eigenvalues of a tridiagonal symmetric matrix between specified boundary indices, using bisection.

### TSTURM
Cray 1 optimized version of the EISPACK routine; finds those eigenvalues of a tridiagonal symmetric matrix which lie in a specified interval and their associated eigenvectors, using bisection and inverse iteration.

---

**SLATEC**

### BAKVEC
Forms eigenvectors of certain real non-symmetric tridiagonal matrix from symmetric tridiagonal matrix output from FIGI.

### BISECT
Computes eigenvalues of symmetric tridiagonal matrix which lie in a 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
Computes eigenvalues and, optionally, eigenvectors of real symmetric tridiagonal matrix.

### RT
Computes eigenvalues and eigenvectors of a special real tridiagonal matrix.

### TINVIT
Eigenvectors of symmetric tridiagonal matrix corresponding to some specified eigenvalues, using inverse iteration.

### TQL1
Computes eigenvalues of symmetric tridiagonal matrix by QL method.

### TQL2
Computes eigenvalues and eigenvectors of symmetric tridiagonal matrix.

### TQLRAT
Computes eigenvalues of symmetric tridiagonal matrix by a rational variant of the QL method.

### TRIDIB
Computes eigenvalues of symmetric tridiagonal matrix which lie in a 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).
F2c2. Eigenvalues and Eigenvectors of Real Symmetric Matrices

**EISPACK**

**IMTQL2**
Finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the implicit QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.

**RS**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a real symmetric matrix.

**RSM**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find all of the eigenvalues and some of the eigenvectors (if desired) of a real symmetric matrix.

**RSP**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a real symmetric packed matrix.

**TQL2**
Finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.

**TRBAK1**
Forms the eigenvectors of a real symmetric matrix by back transforming those of the corresponding symmetric tridiagonal matrix determined by TRED1.

**TRBAK3**
Forms the eigenvectors of a real symmetric matrix by back transforming those of the corresponding symmetric tridiagonal matrix determined by TRED3.

**EISPKD**

**IMTQL2**
Finds the eigenvalues and eigenvectors of a double precision symmetric tridiagonal matrix by the implicit QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.

**RS**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision real symmetric matrix.

**RSM**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and some of the eigenvectors (if desired) of a double precision real symmetric matrix.

**RSP**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision real symmetric packed matrix.

**TQL2**
Finds the eigenvalues and eigenvectors of a double precision symmetric tridiagonal matrix by the QL method. The eigenvectors of a full double precision symmetric matrix can also be found if TRED2 has been used to reduce this full
matrix to tridiagonal form.

**TRBAK1**
Forms the eigenvectors of a double precision real symmetric matrix by back transforming those of the corresponding double precision symmetric tridiagonal matrix determined by TRED1.

**TRBAK3**
Forms the eigenvectors of a double precision real symmetric matrix by back transforming those of the corresponding double precision symmetric tridiagonal matrix determined by TRED3.

**IMSL**

**EHOBKS**
Back transformation to form the eigenvectors of the original symmetric matrix from the eigenvectors of the tridiagonal matrix.

**EIGRS**
Eigenvalues and (optionally) eigenvectors of a real symmetric matrix in symmetric storage mode.

**LOCLIB**

**EIGSFM**
Computes the algebraically smallest or largest $M$ eigenvalues and corresponding eigenvectors of a real symmetric matrix.

**EIGSFS**
Computes eigenvalues within a specified interval and corresponding eigenvectors of a real symmetric matrix.

**NAG**

**F01AZF**
Derives eigenvectors of a real symmetric matrix from eigenvectors of the tridiagonal form produced by F01AYF.

**F02AAF**
Calculates all the eigenvalues of a real symmetric matrix by Householder reduction and the QL algorithm.

**F02ABF**
Calculates all the eigenvalues and eigenvectors of a real symmetric matrix by Householder reduction and the QL algorithm.

**F02AMF**
Calculates all the eigenvalues and eigenvectors of a real symmetric tridiagonal matrix or of a full real symmetric matrix that has been reduced to tridiagonal form using F01AJF.

**F02AVF**
Calculates all the eigenvalues of a real symmetric tridiagonal matrix.

**F02BBF**
Calculates selected eigenvalues and eigenvectors of a real symmetric matrix by Householder reduction, the method of bisection and inverse iteration, where the selected eigenvalues lie between two given values.

**F02BEF**
Calculates selected eigenvalues and eigenvectors of a real symmetric tridiagonal matrix, where the selected eigenvalues lie between two given values.

**F02BFF**
Calculates selected eigenvalues of a real symmetric tridiagonal matrix, where, if the eigenvalues are numbered in ascending order, the numbers of the first and last eigenvalues required are given.
### SCILIB

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMTQL2</strong></td>
<td>Cray 1 optimized version of the EISPACK routine; finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the implicit QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.</td>
</tr>
<tr>
<td><strong>TQL2</strong></td>
<td>Cray 1 optimized version of the EISPACK routine; finds the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the QL method. The eigenvectors of a full symmetric matrix can also be found if TRED2 has been used to reduce this full matrix to tridiagonal form.</td>
</tr>
<tr>
<td><strong>TRBAK1</strong></td>
<td>Cray 1 optimized version of the EISPACK routine; forms the eigenvectors of a real symmetric matrix by back transforming those of the corresponding symmetric tridiagonal matrix determined by TRED1.</td>
</tr>
<tr>
<td><strong>TRBAK3</strong></td>
<td>Cray 1 optimized version of the EISPACK routine; forms the eigenvectors of a real symmetric matrix by back transforming those of the corresponding symmetric tridiagonal matrix determined by TRED3.</td>
</tr>
</tbody>
</table>

### SLATEC

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMTQL2</strong></td>
<td>Computes eigenvalues and eigenvectors of symmetric tridiagonal matrix using implicit QL method.</td>
</tr>
<tr>
<td><strong>RS</strong></td>
<td>Computes eigenvalues and, optionally, eigenvectors of real symmetric matrix.</td>
</tr>
<tr>
<td><strong>RSP</strong></td>
<td>Computes eigenvalues and, optionally, eigenvectors of real symmetric matrix packed into a one-dimensional array.</td>
</tr>
<tr>
<td><strong>SSIEV</strong></td>
<td>Computes the eigenvalues and, optionally, the eigenvectors of a real symmetric matrix.</td>
</tr>
<tr>
<td><strong>SSPEV</strong></td>
<td>Computes the eigenvalues and, optionally, the eigenvectors of a real symmetric matrix stored in packed form.</td>
</tr>
<tr>
<td><strong>TQL2</strong></td>
<td>Computes eigenvalues and eigenvectors of symmetric tridiagonal matrix.</td>
</tr>
<tr>
<td><strong>TRBAK1</strong></td>
<td>Forms the eigenvectors of real symmetric matrix from eigenvectors of symmetric tridiagonal matrix formed by TRED1.</td>
</tr>
<tr>
<td><strong>TRBAK3</strong></td>
<td>Forms eigenvectors of real symmetric matrix from the eigenvectors of symmetric tridiagonal matrix formed by TRED3.</td>
</tr>
</tbody>
</table>
F2c3.  

Eigenvalues and Eigenvectors of Real Non-Symmetric Matrices

EISPACK

BALANC  Balances a real matrix and isolates eigenvalues whenever possible.
BALBAK  Forms the eigenvectors of a real general matrix by back transforming those of the corresponding balanced matrix determined by BALANC.
ELMBAK  Forms the eigenvectors of a real general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by ELMHES.
HQR2    Finds the eigenvalues and eigenvectors of a real upper Hessenberg matrix by the QR method. The eigenvectors of a real general matrix can also be found if ELMHES and ELTRAN or ORTHES and ORTRAN have been used to reduce this general matrix to Hessenberg form and to accumulate the similarity transformations.
ORTBAK  Forms the eigenvectors of a real general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by ORTHES.
RG      Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a real general matrix.

EISPKD

BALANC  Balances a double precision real matrix and isolates eigenvalues whenever possible.
BALBAK  Forms the eigenvectors of a double precision real general matrix by back transforming those of the corresponding double precision balanced matrix determined by BALANC.
ELMBAK  Forms the eigenvectors of a double precision real general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by ELMHES.
HQR2    Finds the eigenvalues and eigenvectors of a double precision real upper Hessenberg matrix by the QR method. The eigenvectors of a double precision real general matrix can also be found if ELMHES and ELTRAN or ORTHES and ORTRAN have been used to reduce this double precision general matrix to Hessenberg form and to accumulate the similarity transformations.
ORTBAK  Forms the eigenvectors of a double precision real general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by ORTHES.
RG      Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision real general matrix.
IMSL

EBALAF Balance a real matrix.
EBBCKF Back transformation of the eigenvectors of a balanced real matrix to form the eigenvectors of the original matrix.
EIGRF Eigenvalues and (optionally) eigenvectors of a real general matrix in full storage mode.

LOCILIB

EIGRG2 Computes selected eigenvectors of a real general matrix.

NAG

F01ALF Transforms eigenvectors of a Hessenberg matrix to those of a real unsymmetric matrix from which the Hessenberg matrix has previously been derived.
F01ATF Balances a real unsymmetric matrix.
F01AUF Transforms eigenvectors of a balanced matrix to those of the original real unsymmetric matrix.
F01LZF Reduces upper triangular matrix to upper bidiagonal form, using pre- and post-multiplication by orthogonal matrices.
F02AFF Calculates all the eigenvalues of a real unsymmetric matrix by reduction to Hessenberg form and the QR algorithm.
F02AGF Calculates all the eigenvalues and eigenvectors of a real unsymmetric matrix by reduction to Hessenberg form and the QR algorithm.
F02AQF Calculates all the eigenvalues and eigenvectors of a real upper Hessenberg matrix or a real unsymmetric matrix which has been reduced to upper Hessenberg form by F01AKF and F01APF and which may have been balanced using F01ATF.
F02BCF Calculates selected eigenvalues and eigenvectors of a real unsymmetric matrix by reduction to Hessenberg form, the QR algorithm and inverse iteration, where the moduli of the selected eigenvalues lie between two given values. Additional precision arithmetic is used to accumulate innerproducts.

NCARLB

EIGRG1 Computes all the eigenvalues and, optionally, all the eigenvectors of a real general matrix.

PORT

DEIGEN Double precision version of EIGEN.
EIGEN Finds all the eigenvalues and eigenvectors of a general real matrix.
SCILIB

BALANC  Cray 1 optimized version of the EISPACK routine; balances a real matrix and isolates eigenvalues whenever possible.

BALBAK  Cray 1 optimized version of the EISPACK routine; forms the eigenvectors of a real general matrix by back transforming those of the corresponding balanced matrix determined by BALANC.

ELMBAK  Cray 1 optimized version of the EISPACK routine; forms the eigenvectors of a real general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by ELMHES.

HQR2  Cray 1 optimized version of the EISPACK routine; finds the eigenvalues and eigenvectors of a real upper Hessenberg matrix by the QR method. The eigenvectors of a real general matrix can also be found if ELMHES and ELTRAN or ORTHES and ORTRAN have been used to reduce this general matrix to Hessenberg form and to accumulate the similarity transformations.

ORTBAK  Cray 1 optimized version of the EISPACK routine; forms the eigenvectors of a real general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by ORTHES.

SLATEC

BALANC  Balances a general real matrix and isolates eigenvalues whenever possible.

BALBAK  Forms eigenvectors of real general matrix from eigenvectors of matrix output from BALANC.

ELMBAK  Forms eigenvectors of real general matrix from eigenvectors of upper Hessenberg matrix output from ELMHES.

HQR2  Computes eigenvalues and eigenvectors of real upper Hessenberg matrix using QR method.

ORTBAK  Forms eigenvectors of general real matrix from eigenvectors of upper Hessenberg matrix output from ORTHES.

RG  Computes eigenvalues and, optionally, eigenvectors of a real general matrix.

SGEEV  Computes the eigenvalues and, optionally, the eigenvectors of a general real matrix.
F2c4. Eigenvalues and Eigenvectors of Hessenberg Matrices

EISPACK

CINVIT  Finds those eigenvectors of a complex upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration.

COMLR  Finds the eigenvalues of a complex upper Hessenberg matrix by the modified LR method.

COMLR2  Finds the eigenvalues and eigenvectors of a complex upper Hessenberg matrix by the modified LR method. The eigenvectors of a complex general matrix can also be found if COMHES has been used to reduce this general matrix to Hessenberg form.

COMQR  Finds the eigenvalues of a complex upper Hessenberg matrix by the QR method.

COMQR2  Finds the eigenvalues and eigenvectors of a complex upper Hessenberg matrix by the QR method. The eigenvectors of a complex general matrix can also be found if CORTH has been used to reduce this general matrix to Hessenberg form.

HQR  Finds the eigenvalues of a real upper Hessenberg matrix by the QR method.

HQR2  Finds the eigenvalues and eigenvectors of a real upper Hessenberg matrix by the QR method. The eigenvectors of a real general matrix can also be found if ELMHES and ELTRAN or ORTHES and ORTRAN have been used to reduce this general matrix to Hessenberg form and to accumulate the similarity transformations.

INVIT  Finds those eigenvectors of a real upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration.

EISPKD

CINVIT  Finds those eigenvectors of a double precision complex upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration.

COMLR  Finds the eigenvalues of a double precision complex upper Hessenberg matrix by the modified LR method.

COMLR2  Finds the eigenvalues and eigenvectors of a double precision complex upper Hessenberg matrix by the modified LR method. The eigenvectors of a double precision complex general matrix can also be found if COMHES has been used to reduce this double precision general matrix to Hessenberg form.

COMQR  Finds the eigenvalues of a double precision complex upper Hessenberg matrix by the QR method.

COMQR2  Finds the eigenvalues and eigenvectors of a double precision complex upper Hessenberg matrix by the QR method. The eigenvectors of a double precision complex general matrix can also be found if CORTH has been used to reduce this double precision general matrix to Hessenberg form.

HQR  Finds the eigenvalues of a double precision real upper Hessenberg matrix by the QR method.

HQR2  Finds the eigenvalues and eigenvectors of a double precision real upper Hessenberg matrix by the QR method. The eigenvectors of a double precision real
general matrix can also be found if ELMHES and ELTRAN or ORTHES and ORTRAN have been used to reduce this double precision general matrix to Hessenberg form and to accumulate the similarity transformations.

INVIT
Finds those eigenvectors of a double precision real upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration.

IMSL

EHBCKF
Back transformation of the eigenvectors of a real upper Hessenberg matrix to form the eigenvectors of the original matrix.

ELRH1C
Eigenvalues of a complex upper Hessenberg matrix.

ELRH2C
Eigenvalues and eigenvectors of a complex upper Hessenberg matrix, and back-transformation of the eigenvectors.

EQRH3F
Eigenvalues and (optionally) eigenvectors of a real upper Hessenberg matrix.

NAG

F02ANF
Calculates all the eigenvalues of a complex upper Hessenberg matrix.

F02APF
Calculates all the eigenvalues of a real upper Hessenberg matrix.

F02BKF
Calculates selected eigenvectors of a real upper Hessenberg matrix.

F02BLF
Calculates selected eigenvectors of a complex upper Hessenberg matrix.

SCILIB

CINVIT
Cray 1 optimized version of the EISPACK routine that finds those eigenvectors of a complex upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration.

COMLR
Cray 1 optimized version of the EISPACK routine that finds the eigenvalues of a complex upper Hessenberg matrix by the modified LR method.

COMLR2
Cray 1 optimized version of the EISPACK routine that finds the eigenvalues and eigenvectors of a complex upper Hessenberg matrix by the modified LR method. The eigenvectors of a complex general matrix can also be found if COMHES has been used to reduce this general matrix to Hessenberg form.

COMQR
Cray 1 optimized version of the EISPACK routine that finds the eigenvalues of a complex upper Hessenberg matrix by the QR method.

COMQR2
Cray 1 optimized version of the EISPACK routine that finds the eigenvalues and eigenvectors of a complex upper Hessenberg matrix by the QR method. The eigenvectors of a complex general matrix can also be found if CORTH has been used to reduce this general matrix to Hessenberg form.

HQR
Cray 1 optimized version of the EISPACK routine; finds the eigenvalues of a real upper Hessenberg matrix by the QR method.

HQR2
Cray 1 optimized version of the EISPACK routine; finds the eigenvalues and eigenvectors of a real upper Hessenberg matrix by the QR method. The eigenvectors of a real general matrix can also be found if ELMHES and ELTRAN or ORTHES and ORTRAN have been used to reduce this general matrix to Hessenberg form and to accumulate the similarity transformations.
Cray 1 optimized version of the EISPACK routine; finds those eigenvectors of a real upper Hessenberg matrix corresponding to specified eigenvalues, using inverse iteration.

**SLATEC**

CINVIT
Computes eigenvectors of a complex upper Hessenberg matrix associated with specified eigenvalues using inverse iteration.

COMLR
Computes eigenvalues of a complex upper Hessenberg matrix using the modified LR method.

COMLR2
Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix using modified LR method.

COMQR
Computes eigenvalues of complex upper Hessenberg matrix using the QR method.

COMQR2
Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix.

HQR
Computes eigenvalues of a real upper Hessenberg matrix using the QR method.

HQR2
Computes eigenvalues and eigenvectors of real upper Hessenberg matrix using QR method.

INVIT
Computes eigenvectors of upper Hessenberg (real) matrix associated with specified eigenvalues by inverse iteration.

---

**EISPACK**

CH
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a complex Hermitian matrix.

HTRIB3
Forms the eigenvectors of a complex Hermitian matrix by back transforming those of the corresponding real symmetric tridiagonal matrix determined by HTRID3.

HTRIBK
Forms the eigenvectors of a complex Hermitian matrix by back transforming those of the corresponding real symmetric tridiagonal matrix determined by HTRIDI.

**EISPKD**

CH
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision complex Hermitian matrix.

HTRIB3
Forms the eigenvectors of a double precision complex Hermitian matrix by back transforming those of the corresponding double precision real symmetric matrix.
tridiagonal matrix determined by HTRID3.

**HTRIBK**
Forms the eigenvectors of a double precision complex Hermitian matrix by back transforming those of the corresponding double precision real symmetric tridiagonal matrix determined by HTRIDI.

**IMSL**

**EIGCH**
Eigenvalues and (optionally) eigenvectors of a complex Hermitian matrix.

**LOCLIB**

**EIGHFS**
Computes eigenvalues within a specified interval and corresponding eigenvectors of a complex, Hermitian matrix.

**NAG**

**F02AWF**
Calculates all the eigenvalues of a complex Hermitian matrix by reduction to real symmetric tridiagonal form and the QL algorithm.

**F02AXF**
Calculates all the eigenvalues and eigenvectors of a complex Hermitian matrix by reduction to real symmetric tridiagonal form and the QL algorithm.

**F02AYF**
Calculates all the eigenvalues and eigenvectors of a complex Hermitian matrix which has been reduced to real symmetric tridiagonal form using F01BCF.

**SCILIB**

**CH**
Cray 1 optimized version of the EISPACK routine that calls the recommended sequence of subroutines from the eigensystem subroutines to find the eigenvalues and eigenvectors (if desired) of a complex Hermitian matrix.

**HTRIB3**
Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a complex Hermitian matrix by back transforming those of the corresponding real symmetric tridiagonal matrix determined by HTRID3.

**HTRIBK**
Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a complex Hermitian matrix by back transforming those of the corresponding real symmetric tridiagonal matrix determined by HTRIDI.

**SLATEC**

**CH**
Computes the eigenvalues and, optionally, eigenvectors of a complex Hermitian matrix.

**CHIEV**
Computes the eigenvalues and, optionally, the eigenvectors of a complex Hermitian matrix.

**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.
F2c6. Eigenvalues and Eigenvectors of Complex Non-Hermitian Matrices

EISPACK

CBABK2 Forms the eigenvectors of a complex general matrix by back transforming those of the corresponding balanced matrix determined by CBAL.

CBAL Balances a complex matrix and isolates eigenvalues whenever possible.

CG Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a complex general matrix.

COMBAK Forms the eigenvectors of a complex general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by COMHES.

COMLR2 Finds the eigenvalues and eigenvectors of a complex upper Hessenberg matrix by the modified LR method. The eigenvectors of a complex general matrix can also be found if COMHES has been used to reduce this general matrix to Hessenberg form.

CORTB Forms the eigenvectors of a complex general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by CORTH.

EISPKD

CBABK2 Forms the eigenvectors of a double precision complex general matrix by back transforming those of the corresponding balanced matrix determined by CBAL.

CBAL Balances a double precision complex matrix and isolates eigenvalues whenever possible.

CG Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision complex general matrix.

COMBAK Forms the eigenvectors of a double precision complex general matrix by back transforming those of the corresponding upper double precision Hessenberg matrix determined by COMHES.

COMLR2 Finds the eigenvalues and eigenvectors of a double precision complex upper Hessenberg matrix by the modified LR method. The eigenvectors of a double precision complex general matrix can also be found if COMHES has been used to reduce this double precision general matrix to Hessenberg form.

CORTB Forms the eigenvectors of a double precision complex general matrix by back transforming those of the corresponding double precision upper Hessenberg matrix determined by CORTH.

IMSL

EBALAC Balance a complex general matrix and isolate eigenvalues whenever possible.

EBBCKC Back transformation of the eigenvectors of a balanced complex matrix to form the eigenvectors of the original matrix.
EIGCC  Eigenvalues and (optionally) eigenvectors of a complex general matrix.

LOCLIB

EIGCG1  Computes all the eigenvalues and, optionally, all the eigenvectors of a general complex matrix.
EIGCG2  Computes selected eigenvectors of a general complex matrix.

NAG

F01ANF  Transforms eigenvectors of a complex upper Hessenberg matrix to those of a complex unsymmetric matrix from which the Hessenberg matrix has previously been derived.
F01AVF  Balances a complex matrix.
F01AWF  Transforms eigenvectors of a balanced matrix to those of the original complex matrix.
F02AJF  Calculates all the eigenvalues of a complex matrix by reduction to upper Hessenberg form and the LR algorithm.
F02AKF  Calculates all the eigenvalues and eigenvectors of a complex matrix by reduction to upper Hessenberg form and the LR algorithm.
F02ARF  Calculates all the eigenvalues and eigenvectors of either a complex upper Hessenberg matrix or a complex full matrix which has been reduced to upper Hessenberg form by F01AMF and may have been balanced using F01AVF.
F02BDF  Calculates selected eigenvalues and eigenvectors of a complex matrix by reduction to Hessenberg form, the LR algorithm and inverse iteration, where the moduli of the selected eigenvalues lie between two given values.

SCILIB

CBAL  Cray 1 optimized version of the EISPACK routine that balances a complex matrix and isolates eigenvalues whenever possible.
CG  Cray 1 optimized version of the EISPACK routine that calls the recommended sequence of subroutines from the eigensystem subroutines to find the eigenvalues and eigenvectors (if desired) of a complex general matrix.
COMBAK  Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a complex general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by COMHES.
COMLR2  Cray 1 optimized version of the EISPACK routine that finds the eigenvalues and eigenvectors of a complex upper Hessenberg matrix by the modified LR method. The eigenvectors of a complex general matrix can also be found if COMHES has been used to reduce this general matrix to Hessenberg form.
CORTB  Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a complex general matrix by back transforming those of the corresponding upper Hessenberg matrix determined by CORTH.
**SLATEC**

CBABK2  Forms eigenvectors of complex general matrix from eigenvectors of matrix output from CBAL.

CBAL    Balances a complex general matrix and isolates eigenvalues whenever possible.

CG      Computes the eigenvalues and, optionally, the eigenvectors of a complex general matrix.

CGEEV   Computes the eigenvalues and, optionally, the eigenvectors of a general complex matrix.

COMBAK  Forms eigenvectors of complex general matrix from eigenvectors of upper Hessenberg matrix output from COMHES.

COMLR2  Computes eigenvalues and eigenvectors of complex upper Hessenberg matrix using modified LR method.

CORTB   Forms eigenvectors of complex general matrix from eigenvectors of upper Hessenberg matrix output from CORTH.

---

**F2c7. Eigenvalues and Eigenvectors of Sparse Matrices**

**LOCLIB**

LASO2   A set of subroutines for computing a few eigenvalues of a large (sparse) symmetric matrix. The author is Dr. David Scott of the Computer Science Department at the University of Texas at Austin.

---

**F2c8. Eigenvalues and Eigenvectors of Banded Matrices**

**EISPACK**

BANDV   Finds those eigenvectors of a real symmetric band matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.

BQR     Finds the eigenvalue of smallest (usually) magnitude of a real symmetric band matrix using the QR algorithm with shifts of origin. Consecutive calls can be made to find further eigenvalues.

RSB     Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a real symmetric band matrix.
**EISPKD**

**BANDV**
Finds those eigenvectors of a double precision real symmetric band matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a double precision symmetric or non-symmetric band coefficient matrix.

**BQR**
Finds the eigenvalue of smallest (usually) magnitude of a double precision real symmetric band matrix using the QR algorithm with shifts of origin. Consecutive calls can be made to find further eigenvalues.

**RSB**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) of a double precision real symmetric band matrix.

**IMSL**

**EIGBS**
Find some eigenvalues and (optionally) eigenvectors of a real symmetric band matrix.

**SCILIB**

**BANDV**
Cray 1 optimized version of the EISPACK routine; finds those eigenvectors of a real symmetric band matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.

**BQR**
Cray 1 optimized version of the EISPACK routine; finds the eigenvalue of smallest (usually) magnitude of a real symmetric band matrix using the QR algorithm with shifts of origin. Consecutive calls can be made to find further eigenvalues.

**SLATEC**

**BANDV**
Forms eigenvectors of real symmetric band matrix associated with a set of ordered approximate eigenvalues by inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.

**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.
F2d. Generalized Eigenvalue Problems: (Ax = lambda Bx, etc.)

**EISPACK**

**QZHES** Accepts a pair of real general matrices and reduces one of them to upper Hessenberg form and the other to upper triangular form using orthogonal transformations. It is usually followed by QZIT, QZVAL and, possibly, QZVEC.

**QZIT** Accepts a pair of real matrices, one of them in upper Hessenberg form and the other in upper triangular form. It reduces the Hessenberg matrix to quasi-triangular form using orthogonal transformations while maintaining the triangular form of the other matrix. It is usually preceded by QZHES and followed by QZVAL and, possibly, QZVEC.

**QZVAL** Accepts a pair of real matrices, one of them in quasi-triangular form and the other in upper triangular form. It reduces the quasi-triangular matrix further, so that any remaining 2X2 blocks correspond to pairs of complex eigenvalues, and returns quantities whose ratios give the generalized eigenvalues. It is usually preceded by QZHES and QZIT and may be followed by QZVEC.

**QZVEC** Accepts a pair of real matrices, one of them in quasi-triangular form (in which each 2X2 block corresponds to a pair of complex eigenvalues) and the other in upper triangular form. It computes the eigenvectors of the triangular problem and transforms the results back to the original coordinate system. It is usually preceded by QZHES, QZIT, and QZVAL.

**REBAK** Forms the eigenvectors of a generalized symmetric eigensystem by back transforming those of the derived symmetric matrix determined by REDUC.

**REBAKB** Forms the eigenvectors of a generalized symmetric eigensystem by back transforming those of the derived symmetric matrix determined by REDUC2.

**REDUC** Reduces the generalized symmetric eigenproblem Ax = lambda Bx, where B is positive definite, to the standard symmetric eigenproblem using the Cholesky factorization of B.

**REDUC2** Reduces the generalized symmetric eigenproblems ABx = lambda x or BAy = lambda y, where B is positive definite, to the standard symmetric eigenproblem using the Cholesky factorization of B.

**RGG** Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) for the real general generalized eigenproblem Ax = lambda Bx.

**RSG** Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) for the real symmetric generalized eigenproblem Ax = lambda Bx.

**RSGAB** Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) for the real symmetric generalized eigenproblem ABx = lambda x.

**RSGBA** Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) for the real symmetric generalized eigenproblem BAx = lambda x.
**EISPKD**

**QZHES**
Accepts a pair of double precision real general matrices and reduces one of them to upper Hessenberg form and the other to upper triangular form using orthogonal transformations. It is usually followed by QZIT, QZVAL and, possibly, QZVEC.

**QZIT**
Accepts a pair of double precision real matrices, one of them in upper Hessenberg form and the other in upper triangular form. It reduces the Hessenberg matrix to quasi-triangular form using orthogonal transformations while maintaining the triangular form of the other matrix. It is usually preceded by QZHES and followed by QZVAL and, possibly, QZVEC.

**QZVAL**
Accepts a pair of double precision real matrices, one of them in quasi-triangular form and the other in upper triangular form. It reduces the quasi-triangular matrix further, so that any remaining 2×2 blocks correspond to pairs of double precision complex eigenvalues, and returns quantities whose ratios give the generalized eigenvalues. It is usually preceded by QZHES and QZIT and may be followed by QZVEC.

**QZVEC**
Accepts a pair of double precision real matrices, one of them in quasi-triangular form (in which each 2×2 block corresponds to a pair of double precision complex eigenvalues) and the other in upper triangular form. It computes the eigenvectors of the triangular problem and transforms the results back to the original coordinate system. It is usually preceded by QZHES, QZIT, and QZVAL.

**REBAK**
Forms the eigenvectors of a generalized double precision symmetric eigensystem by back transforming those of the derived double precision symmetric matrix determined by REDUC.

**REBAKB**
Forms the eigenvectors of a generalized double precision symmetric eigensystem by back transforming those of the derived double precision symmetric matrix determined by REDUC2.

**REDUC**
Reduces the generalized double precision symmetric eigenproblem \( Ax = \lambda Bx \), where \( B \) is positive definite, to the standard double precision symmetric eigenproblem using the Cholesky factorization of \( B \).

**REDUC2**
Reduces the generalized double precision symmetric eigenproblems \( ABx = \lambda x \) or \( BAy = \lambda y \), where \( B \) is positive definite, to the standard double precision symmetric eigenproblem using the Cholesky factorization of \( B \).

**RGG**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) for the double precision real general generalized eigenproblem \( Ax = \lambda Bx \).

**RSG**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) for the double precision real symmetric generalized eigenproblem \( Ax = \lambda Bx \).

**RSGAB**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) for the double precision real symmetric generalized eigenproblem \( ABx = \lambda x \).

**RSGBA**
Calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPKD) to find the eigenvalues and eigenvectors (if desired) for the double precision real symmetric generalized eigenproblem \( BAx = \lambda x \).
Imsl

EIGZC Eigenvalues and (optionally) eigenvectors of the system $Ax = \lambda Bx$ where $A$ and $B$ are complex matrices.

EIGZF Eigenvalues and (optionally) eigenvectors of the system $Ax = \lambda Bx$ where $A$ and $B$ are real matrices.

EIGZS Eigenvalues and (optionally) eigenvectors of the system $Ax = \lambda Bx$ where $A$ and $B$ are real symmetric matrices and $B$ is positive definite.

ELZHC Simultaneously reduce two complex matrices $A$ and $B$ to upper Hessenberg and $B$ to upper triangular form.

ELZVC Calculate the eigenvalues and, optionally, eigenvectors of the system $Ax = \lambda Bx$ where complex matrix $A$ is upper Hessenberg and complex matrix $B$ is upper triangular.

EQZQF Hessenberg reduction for the generalized eigenvalue problem $Ax = \lambda Bx$. Reduction of $A$ to upper Hessenberg form and $B$ to upper triangular form.

EQZTF Explicit QZ iteration for the generalized eigenvalue problem $Ax = \lambda Bx$ where $A$ is in upper Hessenberg form and $B$ is upper triangular. $A$ is reduced to quasi-upper triangular form while keeping matrix $B$ upper triangular.

EQZVF Eigenvalues and optionally, eigenvectors of the generalized eigenvalue problem $Az = \lambda Bz$ where $B$ is upper triangular and $A$ is quasi-upper triangular.

Nag

F01AEF Reduces the generalised eigenproblem $Ax = \lambda Bx$ to the standard symmetric eigenproblem $Pz = \lambda z$, where $A$ is a real symmetric matrix and $B$ is a real symmetric positive definite matrix.

F01AFF Derives eigenvectors $x$ of the eigenproblems $Ax = \lambda Bx$, $ABx = \lambda x$ and $yBA = \lambda y$, where $y$ is the transpose of $x$, from the corresponding eigenvectors $z = Mx$, where $M$ is the transpose of the lower triangular matrix $L$ of the derived standard symmetric eigenproblems. Matrices $A$ and $B$ are real and symmetric. In addition $B$ is positive definite.

F01BDF Reduces the eigenproblems $ABx = \lambda x$, $x^T BA = \lambda x^T$, $BAy = \lambda y$ and $y^T AB = \lambda y^T$ to the standard symmetric eigenproblem $Qz = \lambda z$, where $A$ is a real symmetric matrix and $B$ is a real symmetric positive definite matrix.

F01BEF Derives eigenvectors $y$ of the problems $y^T AB = \lambda y^T$ and $BAy = \lambda y$ from the corresponding eigenvectors of the derived standard symmetric eigenproblem.

F01BVF Transforms the generalised symmetric eigenproblem $Ax = \lambda Bx$ to the equivalent standard eigenproblem $Cy = \lambda y$ where $A$, $B$ and $C$ are symmetric band matrices and $B$ is positive definite. $B$ must have been decomposed by F01BUF.

F02ADF Calculates all the eigenvalues of $A x = \lambda Bx$, where $A$ is a real symmetric matrix and $B$ is a real symmetric positive definite matrix, using Householder reduction and the QL algorithm.

F02AEF Calculates all the eigenvalues and eigenvectors of $Ax = \lambda Bx$, where $A$ is a real symmetric matrix and $B$ is a real symmetric positive definite matrix, using Householder reduction and the QL algorithm.

F02BJF Calculates all the eigenvalues and, if required, all the eigenvectors of the generalized eigenproblem $Ax = \lambda Bx$ where $A$ and $B$ are real, square matrices, using the
QZ algorithm.

F02FHF Finds the eigenvalues of the generalized band symmetric eigenproblem $Ax = \lambda Bx$ where $A$ and $B$ are symmetric band matrices and $B$ is positive definite.

F02FJF Finds eigenvalues and eigenvectors of a real sparse symmetric or generalised symmetric eigenvalue problem.

F02GJF Calculates all the eigenvalues and, if required, all the eigenvectors of the complex generalized eigenproblem $Ax = \lambda Bx$ where $A$ and $B$ are complex, square matrices, using the QZ algorithm.

F02SDF Finds the eigenvector corresponding to a given real eigenvalue for the generalised problem $Ax = \lambda Bx$, or for the standard problem $Ax = \lambda x$, where $A$ and $B$ are real band matrices.

SCILIB

QZHES Cray 1 optimized version of the EISPACK routine that accepts a pair of real general matrices and reduces one of them to upper Hessenberg form and the other to upper triangular form using orthogonal transformations. It is usually followed by QZIT, QZVAL and, possibly, QZVEC.

QZIT Cray 1 optimized version of the EISPACK routine that accepts a pair of real matrices, one of them in upper Hessenberg form and the other in upper triangular form. It reduces the Hessenberg matrix to quasi-triangular form using orthogonal transformations while maintaining the triangular form of the other matrix. It is usually preceded by QZHES and followed by QZVAL and, possibly, QZVEC.

QZVAL Cray 1 optimized version of the EISPACK routine that accepts a pair of real matrices, one of them in quasi-triangular form and the other in upper triangular form. It reduces the quasi-triangular matrix further, so that any remaining $2 \times 2$ blocks correspond to pairs of complex eigenvalues, and returns quantities whose ratios give the generalized eigenvalues. It is usually preceded by QZHES and QZIT and may be followed by QZVEC.

QZVEC Cray 1 optimized version of the EISPACK routine that accepts a pair of real matrices, one of them in quasi-triangular form (in which each $2 \times 2$ block corresponds to a pair of complex eigenvalues) and the other in upper triangular form. It computes the eigenvectors of the triangular problem and transforms the results back to the original coordinate system. It is usually preceded by QZHES, QZIT, and QZVAL.

REBAK Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a generalized symmetric eigensystem by back transforming those of the derived symmetric matrix determined by REDUC.

REBAKB Cray 1 optimized version of the EISPACK routine that forms the eigenvectors of a generalized symmetric eigensystem by back transforming those of the derived symmetric matrix determined by REDUC2.

REDUC Cray 1 optimized version of the EISPACK routine that reduces the generalized symmetric eigenproblem $Ax = \lambda Bx$, where $B$ is positive definite, to the standard symmetric eigenproblem using the Cholesky factorization of $B$.

REDUC2 Cray 1 optimized version of the EISPACK routine that reduces the generalized symmetric eigenproblems $ABx = \lambda x$ or $BAy = \lambda y$, where $B$ is positive definite, to the standard symmetric eigenproblem using the Cholesky factorization of $B$.
RGG
Cray 1 optimized version of the EISPACK routine that calls the recommended sequence of subroutines from the eigensystem subroutines to find the eigenvalues and eigenvectors (if desired) for the real general generalized eigenproblem $Ax = \lambda Bx$.

RSG
Cray 1 optimized version of the EISPACK routine that calls the recommended sequence of subroutines from the eigensystem subroutines to find the eigenvalues and eigenvectors (if desired) for the real symmetric generalized eigenproblem $Ax = \lambda Bx$.

RSGAB
Cray 1 optimized version of the EISPACK routine that calls the recommended sequence of subroutines from the eigensystem subroutines to find the eigenvalues and eigenvectors (if desired) for the real symmetric generalized eigenproblem $ABx = \lambda x$.

RSGBA
Cray 1 optimized version of the EISPACK routine that calls the recommended sequence of subroutines from the eigensystem subroutines to find the eigenvalues and eigenvectors (if desired) for the real symmetric generalized eigenproblem $BAx = \lambda x$.

**SLATEC**

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.

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.

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.

REBAK
Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC.

REBAKB
Forms eigenvectors of generalized symmetric eigensystem from eigenvectors of derived matrix output from REDUC2.

REDUC
Reduces generalized symmetric eigenproblem $Ax = \lambda Bx$, to standard symmetric eigenproblem using Cholesky factorization.

REDUC2
Reduces certain generalized symmetric eigenproblems standard symmetric eigenproblem, using Cholesky factorization.

RGG
Computes eigenvalues and eigenvectors for real generalized eigenproblem: $Ax = \lambda Bx$.

RSG
Computes eigenvalues and, optionally, eigenvectors of symmetric generalized eigenproblem: $Ax = \lambda Bx$. 

Linear Algebra
RSGAB
Computes eigenvalues and, optionally, eigenvectors of symmetric generalized
eigenproblem: \( ABz = \lambda z \).

RSGBA
Computes eigenvalues and, optionally, eigenvectors of symmetric generalized
eigenproblem: \( BAz = \lambda z \).

F4a1. Linear Systems: Real Non-Symmetric Full

**IMSL**

LEQIF
Linear equation solution – full matrices (virtual memory version).

LEQOF
Linear equation solution – full matrices (out-of-core version).

LEQT1F
Linear equation solution – full storage mode – space economizer solution.

LEQT2F
Linear equation solution – full storage mode – high accuracy solution.

LINV1F
Inversion of a matrix – full storage mode – space economizer solution.

LINV2F
Inversion of a matrix – full storage mode – high accuracy solution.

LINV3F
In-place inverse, equation solution, and/or determinant evaluation – full storage mode.

LUDATF
LU decomposition by the Crout Algorithm with optional accuracy test.

LUELMF
Elimination part of solution of \( Ax = b \) (full storage mode).

LUREFF
Refinement of solution to linear equations – full storage mode.

**LINPACK**

SGECO
Factors a real matrix by Gaussian elimination and estimates the condition of the
matrix.

SGEDI
Computes the determinant and inverse of a matrix using the factors computed by
SGECO or SGEFA.

SGEFA
Factors a real matrix by Gaussian elimination.

SGESL
Solves the real system \( Ax = b \) or \( A^T x = b \) using the factors computed by
SGECO or SGEFA.

STRCO
Estimates the condition of a real triangular matrix.

STRDI
Computes the determinant and/or the inverse of an upper or lower real triangular
matrix.

STRSL
Solves systems \( Tx = b \) or \( T^T z = b \) where \( T \) is a triangular matrix of order \( n \).
Here \( T^T \) denotes the transpose of the matrix \( T \).
LOCLIB

SGESLV Computes the solution of a general square linear system.

NAG

F01AAF Calculates the approximate inverse of a real matrix by Crout’s method with partial pivoting.
F01BTF Decomposes a real matrix into a product of triangular matrices LU by Gaussian elimination with partial pivoting. The block-column method used is designed for efficient working on paged virtual machines.
F03AAF Calculates the determinant of a real matrix using the factorisation method of Crout.
F03AFF Decomposes a real matrix into triangular factor matrices LU by Crout’s method and evaluates the determinant.
F04AAF Calculates the approximate solution of a set of real linear equations with multiple right hand sides by Crout’s factorisation method.
F04AEF Calculates the accurate solution of a set of real linear equations with multiple right hand sides by Crout’s factorisation method.
F04AHF Calculates the accurate solution of a set of real linear equations with multiple right hand sides, $AX = B$, where $A$ has been decomposed into triangular matrices using F03AFF.
F04AJF Calculates the approximate solution of a set of real linear equations with multiple right hand sides, $AX = B$, where $A$ has been decomposed into triangular matrices using F03AFF.
F04ARF Calculates the approximate solution of a set of real linear equations with a single right hand side, $Ax = b$, by Crout’s factorisation method.
F04ATF Calculates the accurate solution of a set of real linear equations with a single right hand side, $Ax = b$, by Crout’s factorisation method.
F04AYF Calculates the approximate solution of a set of real linear equations with multiple right hand sides, $AX = B$, where $A$ has been decomposed into triangular matrices using F01BTF.

NCARLB

INVMTX Computes the inverse of a general real matrix using Gaussian elimination with full pivoting.
LINEQSV Solves a real linear system using Gaussian elimination with partial pivoting.
SGEMRGS Solves a general system of linear equations involving many right-hand sides.

PORT

LINEQ Solves a set of linear equations, $AX = B$, where $A$ is an $N$ by $N$ matrix, and $X$ and $B$ are $N$ times $NB$ matrices. A QR decomposition of the coefficient matrix using Householder transformations is followed by a back substitution for the solution.
### SCILIB

**MINV**
Computes determinant and inverse of a square matrix, or solves via Gauss-Jordan elimination several systems of linear equations described by one square matrix and several right-hand sides. This routine has been optimized for the Cray 1.

**SGECO**
Cray 1 optimized version of the LINPACK routine; factors a real matrix by Gaussian elimination and estimates the condition of the matrix.

**SGEDI**
Cray 1 optimized version of the LINPACK routine; computes the determinant and inverse of a matrix using the factors computed by SGECO or SGEFA.

**SGEFA**
Cray 1 optimized version of the LINPACK routine; factors a real matrix by Gaussian elimination.

**SGESL**
Cray 1 optimized version of the LINPACK routine; solves the real system $Ax = b$ or $A^T x = b$ using the factors computed by SGECO or SGEFA.

**STRCO**
Cray 1 optimized version of the LINPACK routine; estimates the condition of a real triangular matrix.

**STRDI**
Cray 1 optimized version of the LINPACK routine; computes the determinant and inverse of a real triangular matrix.

**STRSL**
Cray 1 optimized version of the LINPACK routine; solves systems of the form $Tx = b$ or $T^T x = b$ where $T$ is a triangular matrix of order $n$. Here $T^T$ denotes the transpose of the matrix $T$.

### SLATEC

**SGECO**
Factors a real matrix by Gaussian elimination and estimates the condition number of the matrix.

**SGEDI**
Computes the determinant and inverse of a matrix using the factors computed by SGECO or SGEFA.

**SGEFA**
Factors a real matrix by Gaussian elimination.

**SGEFS**
Solves a general single precision real $n \times n$ system of linear equations.

**SGEIR**
Solves a general single precision real $n \times n$ system of linear equations. Iterative refinement is used to obtain an error estimate.

**SGESL**
Solves the real system $Ax = b$ or $A^T x = b$ using the factors of SGECO or SGEFA.

**STRCO**
Estimates the condition of a real triangular matrix.

**STRDI**
Computes the determinant and inverse of a real triangular matrix.

**STRSL**
Solves systems of the form $Tx = b$ or $T^T x = b$ where $T$ is a triangular matrix of order $N$. 

F4a2. Linear Systems: Real Non-Symmetric Banded

**EISPACK**

**BANDV**
Finds those eigenvectors of a real symmetric band matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.

**EISPKD**

**BANDV**
Finds those eigenvectors of a double precision real symmetric band matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a double precision symmetric or non-symmetric band coefficient matrix.

**IMSL**

**LEQT1B**
Linear equation solution – band storage mode – space economizer solution.

**LEQT2B**
Linear equation solution – band storage mode – high accuracy solution.

**LINPACK**

**SGBCO**
Factors a real band matrix by Gaussian elimination and estimates the condition of the matrix.

**SGBDI**
Computes the determinant of a band matrix using the factors computed by SGBCO or SGBFA. If the inverse is needed, use SGBSL \( n \) times.

**SGBFA**
Factors a real band matrix by elimination.

**SGBSL**
Solves the real band system \( Ax = b \) or \( A^T x = b \) using the factors computed by SGBCO or SGBFA.

**SGTSL**
Given a general tridiagonal matrix and a right hand side, will find the solution.

**LOCLIB**

**CYCRDLLF**
Solves general tridiagonal systems and symmetric tridiagonal systems using cyclic reduction method. Contains diagnostic stubs for entries CRSOL, CRDEC, SCRDEC, SCRSOL, FSCRsol, BSCRsol. These entries now reside in OLDCYCRD.

**OLDCYCRD**
Solves general tridiagonal systems and symmetric tridiagonal systems using cyclic reduction method. There’s a problem with this package: entry SCRDEC contains CFT directive CDIR$ IVDEP which causes bad numbers on COS editions of 1.13 and beyond. Study is underway.
NCAR Software Catalog  

NAG

F01LBF  Decomposes a general real band matrix of order N, with $M_1$ sub-diagonals and $M_2$ super-diagonals, where $M_1 + M_2 + 1$ is much less than N, into triangular matrices using Gaussian elimination with partial pivoting.

F01LEF  Reduces a real tridiagonal matrix to triangular matrices using Gaussian elimination with partial pivoting.

F04EAF  Calculates the approximate solution of a set of real tridiagonal linear equations with a single right hand side.

F04LDF  Calculates the approximate solution of a set of general real band linear equations with multiple right-hand sides $AX = B$ where A has been decomposed into triangular matrices using F01LBF.

F04LEF  Solves a system of tridiagonal equations following the factorization by NAG library routine F01LEF.

NCARLB

BDSLV  Solves banded linear systems.

BGLSLV  BGLSLV computes the solution for a real banded linear system where the upper and lower bandwidths each should be less than 65.

TRDI  Solves a diagonally-dominant tridiagonal linear system using Gaussian elimination without pivoting.

TRIDLA  Solves a diagonally-dominant tridiagonal linear system using Gaussian elimination without pivoting. Extremely fast.

TRISLV  Optimized CAL subroutines to solve $N$ (less than 65) simultaneous tridiagonal linear systems with no pivoting.

SCILIB

BANDV  Cray 1 optimized version of the EISPACK routine; finds those eigenvectors of a real symmetric band matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.

SGBCO  Cray 1 optimized version of the LINPACK routine; factors a real band matrix by Gaussian elimination and estimates the condition of the matrix.

SGBDI  Cray 1 optimized version of the LINPACK routine; computes the determinant of a band matrix using the factors computed by SGBCO or SGBFA. If the inverse is needed, use SGBSL $n$ times.

SGBFA  Cray 1 optimized version of the LINPACK routine; factors a real band matrix by elimination.

SGBSL  Cray 1 optimized version of the LINPACK routine; solves the real band system $Ax = b$ or $A^T x = b$ using the factors computed by SGBCO or SGBFA.

SGTSL  Cray 1 optimized version of the LINPACK routine; given a general tridiagonal matrix and a right hand side, will find the solution.
SLATEC

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANDV</td>
<td>Forms eigenvectors of real symmetric band matrix associated with a set of ordered approximate eigenvalues by inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.</td>
</tr>
<tr>
<td>SGBCO</td>
<td>Factors a real band matrix by Gaussian elimination and estimates the condition number of the matrix.</td>
</tr>
<tr>
<td>SGBDI</td>
<td>Computes the determinant of a band matrix using the factors computed by SGBCO or SGBFA. If the inverse is needed, use SGBSL n times.</td>
</tr>
<tr>
<td>SGBFA</td>
<td>Factors a real band matrix by elimination.</td>
</tr>
<tr>
<td>SGBSL</td>
<td>Solves the real band system $Ax = b$ or $A^T x = b$ using the factors computed by SGBCO or SGBFA.</td>
</tr>
<tr>
<td>SGTSL</td>
<td>Solves the system $Ax = b$ where $a$ is tridiagonal.</td>
</tr>
<tr>
<td>SNBCO</td>
<td>Factors a real band matrix by Gaussian elimination and estimates the condition number.</td>
</tr>
<tr>
<td>SNBDI</td>
<td>Computes the determinant of a band matrix using the factors computed by SNBCO or SNBFA.</td>
</tr>
<tr>
<td>SNBFA</td>
<td>Factors a real band matrix by elimination.</td>
</tr>
<tr>
<td>SNBFS</td>
<td>Solves a general nonsymmetric banded $n \times n$ system of linear equations.</td>
</tr>
<tr>
<td>SNBIR</td>
<td>Solves a general nonsymmetric banded $n \times n$ system of linear equations. Iterative refinement is used to obtain an error estimate.</td>
</tr>
<tr>
<td>SNBSL</td>
<td>Solves a real band system using factors computed by SNBCO or SNBFA.</td>
</tr>
</tbody>
</table>

F4a4. Linear Systems: Real Non-Symmetric Sparse

ITPACK

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCG</td>
<td>Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the Jacobi conjugate gradient method.</td>
</tr>
<tr>
<td>JSI</td>
<td>Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the Jacobi semi-iteration method.</td>
</tr>
<tr>
<td>PERROR</td>
<td>Computes the residual vector for a linear system whose matrix is stored in sparse symmetric or nonsymmetric storage mode. It optionally writes the residual vector, solution vector and an estimate of the number of correct digits in the solutions.</td>
</tr>
<tr>
<td>PRBNDX</td>
<td>Computes the red-black permutation vectors $p$ (and its inverse $ip$ ) if possible.</td>
</tr>
<tr>
<td>RSCG</td>
<td>Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the reduced system conjugate gradient method.</td>
</tr>
</tbody>
</table>
### RSSI
Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the reduced system semi-iteration method.

### SBELM
Removes rows and columns of the matrix where \( \frac{|a_{i,j}|}{a_{i,i}} \leq tol \) for \( j = 1 \) to \( n \) and \( a_{i,i} > 0 \). This is to take care of matrices arising from finite element discretizations of partial differential equations with Dirichlet boundary conditions. Any such rows and corresponding columns are then set to the identity after correcting right hand side.

### SBEND
Restructures the linked list data structure built by SBINI and SBSIJ into the final data structure required by ITPACK. The restructuring can take place in the minimum amount of memory required to hold the nonzero structure of the sparse symmetric matrix but will run quicker if more storage is allowed. SBEND is based on the sparse matrix package SPARSPAK developed by Alan George and Joseph Lui of the University of Waterloo, Waterloo, Ontario.

### SBINI
Initializes the storage arrays IA, JA, IWORK, and A.

### SBSIJ
Aids the construction of the IA, JA, A data structure used in ITPACK. It takes the individual entries of the sparse symmetric or nonsymmetric matrix as given to it at each call via (I,J,VAL) and inserts it into a linked list representation of the sparse symmetric or nonsymmetric matrix.

### SOR
Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the successive overrelaxation method.

### SSORCG
Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the symmetric SOR conjugate gradient method.

### SSORSI
Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the symmetric SOR semi-iteration method.

### LOCLIB

#### ABD

#### BRTD
Computes the solution of a bordered tridiagonal linear system (i.e., tridiagonal except for a last row and column).

#### BTD
Computes the solution of block tridiagonal linear systems based on the standard LU method. Pivoting is performed within a block.

#### BTDMS
Computes the solution of block tridiagonal linear systems based on a minimal storage version of the LU method. Only one row of blocks is required in storage at any time. Partial pivoting is performed within a block.

#### MA32
This package of FORTRAN subroutines from the Harwell Subroutine Library takes advantage of the Cray's vectorization capabilities, and runs only on the Cray. Direct-access datasets are used to save in-core storage.

#### YSMPNSF
A collection of subroutines which are part of the Yale Sparse Matrix Package for solving very large non-symmetric randomly sparse linear systems. This collection has the capability to save information about the matrix decomposition to allow reentry with a new similarly sparse matrix or new right hand side and save considerable computation time. Also see LOCLIB files YSMPNSQ, YSMPOR, and YSMPSF.
YSMPNSQ  A collection of subroutines which are part of the Yale Sparse Matrix Package for solving very large non-symmetric randomly sparse linear systems. This collection is designed to be used when only one large system is to be solved, because no information is saved to allow reentry with another right hand side or similarly structured sparse matrix. Also see LOCLIB files YSMPNSF, YSMPOR, and YSMPSF.

YSMPOR  A collection of subroutines which are part of the Yale Sparse Matrix Package. This collection attempts to optimally reorder the elements of a sparse matrix so that sparse Gaussian Elimination will have minimal fill-in. The use of YSMPOR with the other Yale Sparse Matrix Package files is optional. Also see LOCLIB files YSMPNSF, YSMPNSQ, and YSMPSF.

NAG

F01BRF  Decomposes a real sparse matrix. The routine either forms the LU-decomposition of a permutation of the entire matrix, or, optionally, first permutes the matrix to block lower triangular form and then only decomposes the diagonal blocks.

F01BSF  Decomposes a real sparse matrix using the pivotal sequence previously obtained by F01BRF when a matrix of the same sparsity pattern was decomposed.

F04AXF  Calculates the approximate solution of a set of real sparse linear equations with a single right hand side, $Ax=b$ or $A^T x = b$, where $A$ has been decomposed by F01BRF or F01BSF.

F04QAF  Solves sparse unsymmetric equations, sparse linear least squares problems and sparse damped linear least squares problems using a Lanczos algorithm.

NCARLB

BLKTRI  Solves block tridiagonal linear systems that arise from finite difference approximations to separable two-dimensional elliptic partial differential equations.

TRDIP  Solves a diagonally-dominant periodic tridiagonal linear system using Gaussian elimination without pivoting.

SLATEC

BLKTRI  Solves a block tridiagonal system of linear equations (usually resulting from the discretization of separable two-dimensional elliptic equations).
F4b1a. Linear Systems: Real Symmetric Full Indefinite

IMSL

LEQ1S  Linear equation solution - indefinite matrix - symmetric storage mode - space economizer solution.

LEQ2S  Linear equation solution - indefinite matrix - symmetric storage mode - high accuracy solution.

LINPACK

SSICO  Factors a real symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

SSIDI  Computes the determinant, inertia and inverse of a real symmetric matrix using the factors from SSIFA.

SSIFA  Factors a real symmetric matrix by elimination with symmetric pivoting.

SSISL  Solves the real symmetric system $Ax = b$ using the factors computed by SSIFA.

SSPCO  Factors a real symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

SSPDI  Computes the determinant, inertia and inverse of a real symmetric matrix using the factors from SSPFA, where the matrix is stored in packed form.

SSPFA  Factors a real symmetric matrix stored in packed form by elimination with symmetric pivoting.

SSPSL  Solves the real symmetric system $Ax = b$ using the factors computed by SSPFA.

SCILIB

SSICO  Cray 1 optimized version of the LINPACK routine; factors a real symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

SSIDI  Cray 1 optimized version of the LINPACK routine; computes the determinant, inertia and inverse of a real symmetric matrix using the factors from SSIFA.

SSIFA  Cray 1 optimized version of the LINPACK routine; factors a real symmetric matrix by elimination with symmetric pivoting.

SSISL  Cray 1 optimized version of the LINPACK routine; solves the real symmetric system $Ax = b$ using the factors computed by SSIFA.

SSPCO  Cray 1 optimized version of the LINPACK routine; factors a real symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

SSPDI  Cray 1 optimized version of the LINPACK routine; computes the determinant, inertia and inverse of a real symmetric matrix using the factors from SSPFA, where the matrix is stored in packed form.

SSPFA  Cray 1 optimized version of the LINPACK routine; factors a real symmetric matrix stored in packed form by elimination with symmetric pivoting.
### SSPSL
Cray 1 optimized version of the LINPACK routine; solves the real symmetric system $Ax = b$ using the factors computed by SSPFA.

**SLATEC**

### SSICO
Factors a real symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

### SSIDI
Computes the determinant, inertia and inverse of a real symmetric matrix using the factors from SSIFA.

### SSIFA
Factors a real symmetric matrix by elimination with symmetric pivoting.

### SSISL
Solves the real symmetric system $Ax = b$ using the factors of SSIFA.

### SSPCO
Factors a real symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

### SSPDI
Computes the determinant, inertia, inverse of a real symmetric matrix (packed form) using the factors from SSPFA.

### SSPFA
Factors a real symmetric matrix stored in packed form by elimination with symmetric pivoting.

### SSPSL
Solves the real symmetric system $Ax = b$ using SSPFA.

---

### F4b1b. Linear Systems: Real Symmetric Full Positive-Definite

**IMSL**

### LEQT1P
Linear equation solution – positive definite matrix – symmetric storage mode – space economizer solution.

### LEQT2P
Linear equations solution – positive definite matrix – symmetric storage mode – high accuracy solution.

### LINV1P
Inversion of matrix – positive definite – symmetric storage mode – space economizer solution.

### LINV2P
Inversion of a matrix – positive definite – symmetric storage mode – high accuracy solution.

### LINV3P
In-place inverse, equation solution, positive definite matrix – symmetric storage mode.

### LUDECP
Decomposition of a positive definite matrix – symmetric storage mode.

### LUELMP
Elimination part of the solution of $Ax = b$ – positive definite matrix – symmetric storage mode.

### LUREFP
Refinement of solution to linear equations – positive definite matrix – symmetric storage mode.
**LINPACK**

- **SCHDC** Computes Cholesky factorization of a real positive definite matrix.
- **SPOCO** Factors a real symmetric positive definite matrix and estimates the condition of the matrix.
- **SPODI** Computes the determinant and inverse of a certain real symmetric positive definite matrix (see below) using factors computed by SPOCO, SPOFA or SQRDC.
- **SPOFA** Factors a real symmetric positive definite matrix.
- **SPOSL** Solves the real symmetric positive definite system $Ax = b$ using the factors computed by SPOCO or SPOFA.
- **SPPCO** Factors a real symmetric positive definite matrix stored in packed form and estimates the condition of the matrix.
- **SPPDI** Computes the determinant and inverse of a real symmetric positive definite matrix using the factors computed by SPPCO or SPPFA.
- **SPPFA** Factors a real symmetric positive definite matrix stored in packed form.
- **SPPSL** Solves the real symmetric positive definite matrix stored in packed form.

**NAG**

- **F01ABF** Calculates the accurate inverse of a real symmetric positive definite matrix by Cholesky's method and iterative refinement. (Simplified parameter list).
- **F01ACF** Calculates the accurate inverse of a real symmetric positive definite matrix by Cholesky's method and iterative refinement.
- **F01ADF** Calculates the approximate inverse of a real symmetric positive definite matrix by Cholesky's method.
- **F01BQF** Forms the Cholesky decomposition of a real symmetric matrix $G$ whose lower triangle only is stored. If $G$ is not positive definite, the routine forms the Cholesky decomposition of $G + E$ where $E$ is a diagonal matrix.
- **F01BXF** Performs the Cholesky factorization $U^T U$ of a real symmetric positive definite matrix $A$.
- **F03ABF** Calculates the determinant of a real symmetric positive definite matrix using Cholesky decomposition.
- **F03AEF** Decomposes a real symmetric positive definite matrix $A$ into triangular matrices $LL^T$ using Cholesky's method, and evaluates the determinant.
- **F03AGF** Decomposes a real symmetric positive definite band matrix into triangular matrices using Cholesky's method and evaluates the determinant.
- **F04ABF** Calculates the accurate solution of a set of real symmetric positive definite linear equations with multiple right hand sides, by Cholesky's decomposition method.
- **F04AFF** Calculates the accurate solution of a set of real symmetric positive definite linear equations with multiple right hand sides, $AX = B$, where $A$ has been decomposed into triangular matrices $A = LL^T$ using F03AEF.
- **F04AGF** Calculates the approximate solution of a set of real symmetric positive definite linear equations with multiple right hand sides, $AX = B$, where $A$ has been...
decomposed into triangular matrices $A = LL^T$ using F03AEF.

**F04AQF**
Calculates the approximate solution of a set of real symmetric positive definite linear equations with a single right hand side $Ax = b$ where $A$ has been decomposed into $LDL^T$ using F01BFF. (Economical storage).

**F04ASF**
Calculates the accurate solution of a set of real symmetric positive definite linear equations with a single right hand side, $Ax = b$, by Cholesky’s decomposition method.

**F04AZF**
Calculates the approximate solution of a set of real symmetric positive definite linear equations with multiple right hand sides, $AX = B$, where $A$ has been decomposed into triangular matrices using F01BXF.

**NCARLB**

**CHLSLV**
Solves a real, symmetric, positive definite linear system by Cholesky decomposition.

**SPDSLV**
Computes the solution for a real symmetric positive definite linear system.

**SCILIB**

**SCHDC**
Cray 1 optimized version of the LINPACK routine; computes Cholesky decomposition of a real positive definite matrix.

**SPOCO**
Cray 1 optimized version of the LINPACK routine; factors a real symmetric positive definite matrix and estimates the condition of the matrix.

**SPODI**
Cray 1 optimized version of the LINPACK routine; computes the determinant and inverse of a certain real symmetric positive definite matrix (see below) using factors computed by SPOCO, SPOFA or SQRDC.

**SPOFA**
Cray 1 optimized version of the LINPACK routine; factors a real symmetric positive definite matrix.

**SPOSOL**
Cray 1 optimized version of the LINPACK routine; solves the real symmetric positive definite system $Ax = b$ using the factors computed by SPOCO or SPOFA.

**SPPCO**
Cray 1 optimized version of the LINPACK routine; factors a real symmetric positive definite matrix stored in packed form and estimates the condition of the matrix.

**SPPDI**
Cray 1 optimized version of the LINPACK routine; computes the determinant and inverse of a real symmetric positive definite matrix using the factors computed by SPPCO or SPPFA.

**SPPFA**
Cray 1 optimized version of the LINPACK routine; factors a real symmetric positive definite matrix stored in packed form.

**SPPSL**
Cray 1 optimized version of the LINPACK routine; solves the real symmetric positive definite system $Ax = b$ using the factors computed by SPPCO or SPPFA.
SLATEC

SCHDC Computes the Cholesky decomposition of a positive definite matrix. A pivoting option allows the user to estimate the condition of a positive definite matrix or determine the rank of a positive semidefinite matrix.

SPOCO Factors a real symmetric positive definite matrix and estimates the condition number of the matrix.

SPODI Computes the determinant and inverse of a certain real symmetric positive definite matrix using the factors computed by SPOCO, SPOFA or SQRDC.

SPOFA Factors a real symmetric positive definite matrix.

SPOFS Solves a positive definite symmetric real $n \times n$ system of linear equations.

SPOIR Solves a positive definite symmetric real $n \times n$ system of linear equations. Iterative refinement is used to obtain an error estimate.

SPOS Solves the real symmetric positive definite system $Ax = b$ using the factors computed by SPOCO or SPOFA.

SPPCO Factors a real symmetric positive definite matrix stored in packed form and estimates the condition of the matrix.

SPPDI Computes the determinant and inverse of a real symmetric positive definite matrix using factors from SPPCO or SPPFA.

SPPFA Factors a real symmetric positive definite matrix (packed form).

SPPSL Solves the real symmetric positive definite system $Ax = b$ using the factors computed by SPPCO or SPPFA.

F4b2. Linear Systems: Real Symmetric Banded

IMSL

LEQ1PB Linear equation solution – positive definite symmetric band matrix – band symmetric storage mode – space economizer solution.

LEQ2PB Linear equation solution – positive definite band symmetric matrix – band symmetric storage mode – high accuracy solution.

LIN1PB Inversion of a matrix – positive definite band symmetric matrix – band symmetric storage mode – space economizer solution.

LIN2PB Inversion of matrix – positive definite band symmetric matrix – band symmetric storage mode – high accuracy solution.

LUDAPB Decomposition of a positive definite band symmetric matrix – band symmetric storage mode.

LUELPB Elimination part of solution of $Ax = b$ – positive definite band symmetric matrix – band symmetric storage mode.
LUREPB Refinement of solution to linear equations – positive definite band symmetric matrix – band symmetric storage mode.

**LINPACK**

**SPBCO** Factors a real symmetric positive definite matrix stored in band form and estimates the condition of the matrix.

**SPBDI** Computes the determinant of a real symmetric positive definite band matrix using the factors computed by SPBCO or SPBFA. If the inverse is needed, use SPBSL \( n \) times.

**SPBFA** Factors a real symmetric positive definite matrix stored in band form.

**SPBSL** Solves the real symmetric positive definite band system \( Ax = b \) using the factors computed by SPBCO or SPBFA.

**SPTSL** Given a positive definite tridiagonal matrix and a right hand side, will find the solution.

**LOCLIB**

**CYCRDLLF** Solves general tridiagonal systems and symmetric tridiagonal systems using cyclic reduction method. Contains diagnostic stubs for entries CRSOL, CRDEC, SCRDEC, SCRSDOL, FSCRSDOL, BSCRSDOL. These entries now reside in OLD-CYCRD.

**OLDCYCRD** Solves general tridiagonal systems and symmetric tridiagonal systems using cyclic reduction method. There's a problem with this package: entry SCRDEC contains CFT directive CDIR$ IVDEP which causes bad numbers on COS editions of 1.13 and beyond. Study is underway.

**NAG**

**F01MCF** Computes the Cholesky factorisation of a symmetric, positive-definite, variable-bandwidth matrix.

**F03ACF** Calculates the determinant of a real symmetric positive definite band matrix using Cholesky decomposition.

**F04ACF** Calculates the approximate solution of a set of real symmetric positive definite band equations with multiple right hand sides by Cholesky’s decomposition method.

**F04ALF** Calculates the approximate solution of a set of real symmetric positive definite band linear equations with multiple right hand sides, \( AX = B \), where \( A \) has been decomposed into triangular matrices using F03AGF.

**F04FAF** Calculates the approximate solution of a set of real symmetric positive definite tridiagonal linear equations.

**F04MCF** Computes the approximate solution of a system of real linear equations with multiple right hand sides, \( Ax = B \), where \( A \) is a symmetric positive-definite variable-bandwidth matrix, which has previously been factorised by F01MCF. Related systems may also be solved.
NCARLB

BSPSLV Computes the solution for a real banded symmetric positive definite matrix where the bandwidth must be less than 65.

SCILIB

SPBCO Cray 1 optimized version of the LINPACK routine; factors a real symmetric positive definite matrix stored in band form and estimates the condition of the matrix.

SPBDI Cray 1 optimized version of the LINPACK routine; computes the determinant of a real symmetric positive definite band matrix using the factors computed by SPBCO or SPBFA. If the inverse is needed, use SPBSL \( n \) times.

SPBFA Cray 1 optimized version of the LINPACK routine; factors a real symmetric positive definite matrix stored in band form.

SPBSL Cray 1 optimized version of the LINPACK routine; solves the real symmetric positive definite band system \( Ax = b \) using the factors computed by SPBCO or SPBFA.

SPTSL Cray 1 optimized version of the LINPACK routine; given a positive definite tridiagonal matrix and a right hand side, will find the solution.

SLATEC

SPBCO Factors a real symmetric positive definite matrix (band form) and estimates the condition of the matrix.

SPBDI Computes the determinant of a real symmetric positive definite band matrix using the factors computed by SPBCO or SPBFA. If the inverse is needed, use SPBSL \( n \) times.

SPBFA Factors a real symmetric positive definite matrix (band form).

SPBSL Solves the real symmetric positive definite band system \( Ax = b \) using the factors computed by SPBCO or SPBFA.

SPTSL Solves the system \( Ax = b \) where \( A \) is positive definite and tridiagonal.

F4b4. Linear Systems: Real Symmetric Sparse

ITPACK

JCG Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the Jacobi conjugate gradient method.

JSI Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the Jacobi semi-iteration method.

PERROR Computes the residual vector for a linear system whose matrix is stored in sparse symmetric or nonsymmetric storage mode. It optionally writes the residual.
PRBNDX: Computes the red-black permutation vectors \( p \) (and its inverse \( ip \)) if possible.

RSCG: Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the reduced system conjugate gradient method.

RSSI: Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the reduced system semi-iteration method.

SBELM: Removes rows and columns of the matrix where \( \frac{|a_{i,j}|}{a_{i,i}} \leq tol \) for \( j = 1 \) to \( n \) and \( a_{i,i} > 0 \). This is to take care of matrices arising from finite element discretizations of partial differential equations with Dirichlet boundary conditions. Any such rows and corresponding columns are then set to the identity after correcting right hand side.

SBEND: Restructures the linked list data structure built by SBINI and SBSIJ into the final data structure required by ITPACK. The restructuring can take place in the minimum amount of memory required to hold the nonzero structure of the sparse symmetric matrix but will run quicker if more storage is allowed. SBEND is based on the sparse matrix package SPARSPAK developed by Alan George and Joseph Lui of the University of Waterloo, Waterloo, Ontario.

SBINI: Initializes the storage arrays IA, JA, IWORK, and A.

SBSIJ: Aids the construction of the IA, JA, A data structure used in ITPACK. It takes the individual entries of the sparse symmetric or nonsymmetric matrix as given to it at each call via \((I,J,VAL)\) and inserts it into a linked list representation of the sparse symmetric or nonsymmetric matrix.

SOR: Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the successive overrelaxation method.

SSORCG: Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the symmetric SOR conjugate gradient method.

SSORSI: Iteratively computes the solution for a real sparse symmetric or nonsymmetric linear system using the symmetric SOR semi-iteration method.

LOCLIB

YSMPSF: A collection of subroutines which are part of the Yale Sparse Matrix Package for solving very large randomly sparse symmetric positive definite linear systems. This collection has the capability to save information about the matrix decomposition to allow reentry with a new similarly sparse matrix or new right hand side and save considerable computation time. Also see LOCLIB files YSMPSF, YSMPSNSQ, and YSMPOR.

NAG

F01MAF: Finds an incomplete Cholesky factorisation of a sparse symmetric positive-definite matrix.

F04MAF: Solves a sparse symmetric positive definite system of linear equations, \( Ax = b \), using preconditioned conjugate gradient method, following the factorization of \( A \) by NAG library routine F01MAF.
F04MBF Solves a system of sparse symmetric linear equations using a Lanczos algorithm.

F4c1. Linear Systems: Complex Non-Hermitian Full

IMSL

LEQ2C Linear equation solution – complex matrix – high accuracy solution.
LEQT1C Matrix decomposition, linear equation solution – space economizer solution – complex matrices.

LINPACK

CGECO Factors a complex matrix by Gaussian elimination and estimates the condition of the matrix.
CGED1 Computes the determinant and inverse of a complex matrix using the factors computed by CGECO or CGEFA.
CGEFA Factors a complex matrix by Gaussian elimination.
CGESL Solves the complex system $Ax = b$ or $A^H x = b$ using the factors computed by CGECO or CGEFA.
CTRRO Estimates the condition of a complex triangular matrix.
CTRDI Computes the determinant and/or the inverse of an upper or lower complex triangular matrix.
CTRSL Solves complex systems $Tx = b$ or $T^H x = b$ where $T$ is a triangular matrix of order $n$. Here $T^H$ denotes the conjugate transpose of the matrix $T$.

NAG

F08ADF Calculates the determinant of a complex matrix using the factorisation method of Crout.
F03AHF Decomposes a complex matrix into triangular factor matrices LU by Crout’s method and evaluates the determinant.
F04ADF Calculates the approximate solution of a set of complex linear equations with multiple right hand sides by Crout’s factorisation method.
F04AKF Calculates the approximate solution of a set of complex linear equations with multiple right hand sides $AX = B$ where $A$ has been decomposed into triangular matrices using F03AHF.
**PORT**

**CLINQ**
Solves a set of complex linear equations with one or more right hand sides. A QR decomposition with Householder transformations is used.

**DCLINQ**
Double precision version of CLINQ.

**SCILIB**

**CGECO**
Cray 1 optimized version of the LINPACK routine that factors a complex matrix by Gaussian elimination and estimates the condition of the matrix.

**CGEDI**
Cray 1 optimized version of the LINPACK routine that computes the determinant and inverse of a complex matrix using the factors computed by CGECO or CGEFA.

**CGEFA**
Cray 1 optimized version of the LINPACK routine that factors a complex matrix by Gaussian elimination.

**CGESL**
Cray 1 optimized version of the LINPACK routine that solves the complex system $Ax = b$ or $A^H x = b$ using the factors computed by CGECO or CGEFA.

**CTRCO**
Cray 1 optimized version of the LINPACK routine that estimates the condition of a complex triangular matrix.

**CTRDI**
Cray 1 optimized version of the LINPACK routine that computes the determinant and/or the inverse of an upper or lower complex triangular matrix.

**CTRSL**
Cray 1 optimized version of the LINPACK routine that solves complex systems $Tx = b$ or $T^H x = b$ where $T$ is a triangular matrix of order $n$. Here $T^H$ denotes the conjugate transpose of the matrix $T$.

**SLATEC**

**CGECO**
Factors a complex matrix by Gaussian elimination and estimates the condition of the matrix.

**CGEDI**
Computes the determinant and inverse of a complex matrix using the factors computed by CGECO or CGEFA.

**CGEFA**
Factors a complex matrix by Gaussian elimination.

**CGEFS**
CGEFS solves a general complex $n \times n$ system of linear equations.

**CGEIR**
Solves a general complex $n \times n$ system of linear equations. Iterative refinement is used to obtain an error estimate.

**CGESL**
Solves the complex system $Ax = b$ or $A^H x = b$ using the factors computed by CGECO or CGEFA.

**CTRCO**
Estimates the condition of a complex triangular matrix.

**CTRDI**
Computes the determinant and inverse of a complex triangular matrix.

**CTRSL**
Solves systems of the form $Tx = b$ or $T^H x = b$ where $T$ is a triangular matrix. Here $T^H$ is conjugate transpose.
F4c2. Linear Systems: Complex Non-Hermitian Banded

**LINPACK**

- **CGBCO** Factors a complex band matrix by Gaussian elimination and estimates the condition of the matrix.
- **CGBDI** Computes the determinant of a complex band matrix using the factors computed by CGBCO or CGBFA. If the inverse is needed, use CGBSL \( n \) times.
- **CGBFA** Factors a complex band matrix by elimination.
- **CGBSL** Solves the complex band system \( Ax = b \) or \( A^H x = b \) using the factors computed by CGBCO or CGBFA.
- **CGTSL** Given a general complex tridiagonal matrix and a right hand side, will find the solution.

**NAG**

- **F01NAF** Finds the LU factorisation of a complex band matrix by Gaussian elimination with partial pivoting.
- **F04NAF** Solves a system of complex band equations following the factorization by NAG library routine F01NAF.

**SCILIB**

- **CGBCO** Cray 1 optimized version of the LINPACK routine that factors a complex band matrix by Gaussian elimination and estimates the condition of the matrix.
- **CGBDI** Cray 1 optimized version of the LINPACK routine that computes the determinant of a complex band matrix using the factors computed by CGBCO or CGBFA. If the inverse is needed, use CGBSL \( n \) times.
- **CGBFA** Cray 1 optimized version of the LINPACK routine that factors a complex band matrix by elimination.
- **CGBSL** Cray 1 optimized version of the LINPACK routine that solves the complex band system \( Ax = b \) or \( A^H x = b \) using the factors computed by CGBCO or CGBFA.
- **CGTSL** Cray 1 optimized version of the LINPACK routine that given a general complex tridiagonal matrix and a right hand side, will find the solution.

**SLATEC**

- **CGBCO** Factors a complex band matrix by Gaussian elimination and estimates the condition of the matrix.
- **CGBDI** Computes the determinant of a complex band matrix using factors from CGBCO or CGBFA. If the inverse is needed, use CGBSL \( n \) times.
- **CGBFA** Factors a complex band matrix by elimination.
- **CGBSL** Solves the complex band system \( Ax = b \) or \( A^H x = b \) using the factors computed by CGBCO or CGBFA.
CGTSL  Solves a general complex tridiagonal system of equations.
CNBCO  Factors a complex band matrix by Gaussian elimination and estimates the condition of the matrix.
CNBDI  Computes the determinant of a complex band matrix using factors computed by CNBCO or CNBFA.
CNBFA  Factors a complex band matrix by elimination.
CNBFS  Solves a general nonsymmetric banded complex $n \times n$ system of linear equations.
CNBIR  Solves a general nonsymmetric banded complex $n \times n$ system of linear equations. Iterative refinement is used to obtain an error estimate.
CNBSL  Solves a complex band system using factors computed by CNBCO or CNBFA.

F4d1a. Linear Systems: Complex Hermitian Full Indefinite

LINPACK

CHICO  Factors a complex Hermitian matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
CHIDI  Computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHIFA.
CHIFA  Factors a complex Hermitian matrix by elimination with symmetric pivoting.
CHISL  Solves the complex Hermitian system $Ax = b$ using the factors computed by CHIFA.
CHPCO  Factors a complex Hermitian matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.
CHPDI  Computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHPFA, where the matrix is stored in packed form.
CHPFA  Factors a complex Hermitian matrix stored in packed form by elimination with symmetric pivoting.
CHPSL  Solves the complex Hermitian system $Ax = b$ using the factors computed by CHPFA.
CSICO  Factors a complex symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
CSIDI  Computes the determinant and inverse of a complex symmetric matrix using the factors from CSIFA.
CSIFA  Factors a complex symmetric matrix by elimination with symmetric pivoting.
CSISL  Solves the complex symmetric system $Ax = b$ using the factors computed by CSIFA.
CSPCO  Factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.
CSPDI Computes the determinant and inverse of a complex symmetric matrix using the factors from CSPFA, where the matrix is stored in packed form.

CSPFA Factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting.

CSPSL Solves the complex symmetric system $Ax = b$ using the factors computed by CSPFA.

**SCILIB**

CHICO Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

CHIDI Cray 1 optimized version of the LINPACK routine that computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHIFA.

CHIFA Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian matrix by elimination with symmetric pivoting.

CHISL Cray 1 optimized version of the LINPACK routine that solves the complex Hermitian system $Ax = b$ using the factors computed by CHIFA.

CHPCO Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

CHPDI Cray 1 optimized version of the LINPACK routine that computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHPFA, where the matrix is stored in packed form.

CHPFA Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian matrix stored in packed form by elimination with symmetric pivoting.

CHPSL Cray 1 optimized version of the LINPACK routine that solves the complex Hermitian system $Ax = b$ using the factors computed by CHPFA.

CSICO Cray 1 optimized version of the LINPACK routine that factors a complex symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

CSIDI Cray 1 optimized version of the LINPACK routine that computes the determinant and inverse of a complex symmetric matrix using the factors from CSIFA.

CSIFA Cray 1 optimized version of the LINPACK routine that factors a complex symmetric matrix by elimination with symmetric pivoting.

CSISL Cray 1 optimized version of the LINPACK routine that solves the complex symmetric system $Ax = b$ using the factors computed by CSIFA.

CSPCO Cray 1 optimized version of the LINPACK routine that factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

CSPDI Cray 1 optimized version of the LINPACK routine that computes the determinant and inverse of a complex symmetric matrix using the factors from CSPFA, where the matrix is stored in packed form.

CSPFA Cray 1 optimized version of the LINPACK routine that factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting.

**Linear Algebra**

Category F4d1a
<table>
<thead>
<tr>
<th>CSPSL</th>
<th>Cray 1 optimized version of the LINPACK routine that solves the complex symmetric system $Ax = b$ using the factors computed by CSPFA.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHICO</td>
<td>Factors a complex Hermitian matrix by elimination with symmetric pivoting and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CHIDI</td>
<td>Computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHIFA.</td>
</tr>
<tr>
<td>CHIFA</td>
<td>Factors a complex Hermitian matrix by elimination (symmetric pivoting).</td>
</tr>
<tr>
<td>CHISL</td>
<td>Solves the complex Hermitian system $Ax = b$ using factors of CHIFA.</td>
</tr>
<tr>
<td>CHPCO</td>
<td>Factors a complex Hermitian matrix (packed form) by elimination with symmetric pivoting and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CHPDI</td>
<td>Computes the determinant, inertia and inverse of a complex Hermitian matrix (packed form) using the factors from CHIFA.</td>
</tr>
<tr>
<td>CHPFA</td>
<td>Factors a complex Hermitian matrix (packed form) by elimination with symmetric pivoting.</td>
</tr>
<tr>
<td>CHPSL</td>
<td>Solves the complex Hermitian system $Ax = b$ using factors of CHPFA.</td>
</tr>
<tr>
<td>CSICO</td>
<td>Factors a complex symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CSIDI</td>
<td>Computes the determinant and inverse of a complex symmetric matrix using the factors from CSIFA.</td>
</tr>
<tr>
<td>CSIFA</td>
<td>Factors a complex symmetric matrix by elimination (symmetric pivoting).</td>
</tr>
<tr>
<td>CSISL</td>
<td>Solves the complex symmetric system $Ax = b$ using factors from CSIFA.</td>
</tr>
<tr>
<td>CSPCO</td>
<td>Factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CSPDI</td>
<td>Computes the determinant and inverse of a complex symmetric matrix stored in packed form using factors from CSPFA.</td>
</tr>
<tr>
<td>CSPFA</td>
<td>Factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting.</td>
</tr>
<tr>
<td>CSPSL</td>
<td>Solves the complex symmetric system $Ax = b$ using factors from CSPFA.</td>
</tr>
</tbody>
</table>
### F4d1b. Linear Systems: Complex Hermitian Full Positive-Definite

#### LINPACK

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCHDC</td>
<td>Computes Cholesky factorization of a complex positive definite matrix.</td>
</tr>
<tr>
<td>CPOCO</td>
<td>Factors a complex Hermitian positive definite matrix and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CPODI</td>
<td>Computes the determinant and inverse of a certain complex Hermitian positive definite matrix using the factors computed by CPOCO, CPOFA or CQRDC.</td>
</tr>
<tr>
<td>CPOFA</td>
<td>Factors a complex Hermitian positive definite matrix.</td>
</tr>
<tr>
<td>CPOSF</td>
<td>Solves the complex Hermitian positive definite system $Ax=b$ using the factors computed by CPOCO or CPOFA.</td>
</tr>
<tr>
<td>CPPCO</td>
<td>Factors a complex Hermitian positive definite matrix stored in packed form and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CPPDI</td>
<td>Computes the determinant and inverse of a complex Hermitian positive definite matrix using the factors computed by CPPCO or CPPFA.</td>
</tr>
<tr>
<td>CPPFA</td>
<td>Factors a complex Hermitian positive definite matrix stored in packed form.</td>
</tr>
<tr>
<td>CPPSL</td>
<td>Solves the complex Hermitian positive definite system $Ax=b$ using the factors computed by CPPCO or CPPFA.</td>
</tr>
</tbody>
</table>

#### NAG

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01BNF</td>
<td>Performs the Cholesky decomposition of a complex positive definite Hermitian matrix, given the lower triangle of the matrix.</td>
</tr>
<tr>
<td>F01BPF</td>
<td>Determines the inverse of a complex positive definite Hermitian matrix, given the lower triangle of the matrix.</td>
</tr>
<tr>
<td>F03AMF</td>
<td>Finds the determinant of a complex positive definite Hermitian matrix $A$, following a Cholesky decomposition of $A$ as given by F01BNF.</td>
</tr>
<tr>
<td>F04AWF</td>
<td>Calculates the approximate solutions of a set of complex linear equations with multiple right hand sides, $AX=B$, where $A$ is positive definite Hermitian, following the Cholesky decomposition of $A$ by F01BNF.</td>
</tr>
</tbody>
</table>

#### SCILIB

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCHDC</td>
<td>Cray 1 optimized version of the LINPACK routine that computes Cholesky factorization of a complex positive definite matrix.</td>
</tr>
<tr>
<td>CPOCO</td>
<td>Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian positive definite matrix and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>CPODI</td>
<td>Cray 1 optimized version of the LINPACK routine that computes the determinant and inverse of a certain complex Hermitian positive definite matrix using the factors computed by CPOCO, CPOFA or CQRDC.</td>
</tr>
<tr>
<td>CPOFA</td>
<td>Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian positive definite matrix.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| CPOSL     | Cray 1 optimized version of the LINPACK routine that solves the complex Her-
  | mitian positive definite system $Ax = b$ using the factors computed by CPOCO or |
| CPPCO     | Cray 1 optimized version of the LINPACK routine that factors a complex Her-
  | mitian positive definite matrix stored in packed form and estimates the condi-
  | tion of the matrix.                                                          |
| CPPDI     | Cray 1 optimized version of the LINPACK routine that computes the determi-
  | nant and inverse of a complex Hermitian positive definite matrix using the fac-
  | tors computed by CPPCO or CPPFA.                                             |
| CPPFA     | Cray 1 optimized version of the LINPACK routine that factors a complex Her-
  | mitian positive definite matrix stored in packed form.                       |
| CPPPSL    | Cray 1 optimized version of the LINPACK routine that solves the complex Her-
  | mitian positive definite system $Ax = b$ using the factors computed by CPPCO or |
| CPPSL     | Cray 1 optimized version of the LINPACK routine that solves the complex Her-
  | mitian positive definite system $Ax = b$ using the factors computed by CPPCO or |

**SLATEC**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCHDC</td>
<td>Computes the Cholesky decomposition of a positive definite matrix. A pivoting</td>
</tr>
<tr>
<td>option allows the user to estimate the condition of a positive definite matrix or</td>
<td></td>
</tr>
<tr>
<td>determine the rank of a positive definite matrix.</td>
<td></td>
</tr>
</tbody>
</table>
| CPOCO     | Factors a complex Hermitian positive definite matrix and estimates the condi-
  | tion of the matrix.                                                          |
| CPODI     | Computes the determinant and inverse of a certain complex Hermitian positive |
  | definite matrix using factors of CPOCO, CPOFA, or CQRDC.                     |
| CPOFA     | Factors a complex Hermitian positive definite matrix.                        |
| CPOFS     | Solves a positive definite symmetric complex $n \times n$ system of linear eq-
  | uations.                                                                     |
| CPOIR     | Solves a positive definite Hermitian complex $n \times n$ system of linear eq-
  | uations. Iterative refinement is used to obtain an error estimate.           |
| CPOSL     | Solves the complex Hermitian positive definite system $Ax = b$ using the fac-
  | tors computed by CPOCO or CPOFA.                                             |
| CPPCO     | Factors a complex Hermitian positive definite matrix stored in packed form and |
  | estimates the condition of the matrix.                                       |
| CPPDI     | Computes the determinant and inverse of a complex Hermitian positive definite |
  | matrix using factors from CPPCO or CPPFA.                                    |
| CPPFA     | Factors a complex Hermitian positive definite matrix (packed form).          |
| CPPPSL    | Solves the complex Hermitian positive definite system $Ax = b$ using the fac-
  | tors computed by CPPCO or CPPFA.                                             |
F4d2. Linear Systems: Complex Hermitian Banded

**LINPACK**

- **CPBCO**
  - Factors a complex Hermitian positive definite matrix stored in band form and estimates the condition of the matrix.

- **CPBDI**
  - Computes the determinant of a complex Hermitian positive definite band matrix using the factors computed by CPBCO or CPBFA. If the inverse is needed, use CPBSL \( n \) times.

- **CPBFA**
  - Factors a complex Hermitian positive definite matrix stored in band form.

- **CPBSL**
  - Solves the complex Hermitian positive definite band system \( Ax = b \) using the factors computed by CPBCO or CPBFA.

- **CPTSL**
  - Given a complex positive definite tridiagonal matrix and a right hand side, will find the solution.

**SCILIB**

- **CPBCO**
  - Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian positive definite matrix stored in band form and estimates the condition of the matrix.

- **CPBDI**
  - Cray 1 optimized version of the LINPACK routine that computes the determinant of a complex Hermitian positive definite band matrix using the factors computed by CPBCO or CPBFA. If the inverse is needed, use CPBSL \( n \) times.

- **CPBFA**
  - Cray 1 optimized version of the LINPACK routine that factors a complex Hermitian positive definite matrix stored in band form.

- **CPBSL**
  - Cray 1 optimized version of the LINPACK routine that solves the complex Hermitian positive definite band system \( Ax = b \) using the factors computed by CPBCO or CPBFA.

- **CPTSL**
  - Cray 1 optimized version of the LINPACK routine that given a complex positive definite tridiagonal matrix and a right hand side, will find the solution.

**SLATEC**

- **CPBCO**
  - Factors a complex Hermitian positive definite matrix stored in band form and estimates the condition of the matrix.

- **CPBDI**
  - Computes the determinant of a complex Hermitian positive definite band matrix using factors from CPBCO or CPBFA. Use CPBSL for inverse using the factors computed by CPBCO or CPBFA.

- **CPBFA**
  - Factors a complex Hermitian positive definite matrix (band form).

- **CPBSL**
  - Solves the complex Hermitian positive definite band system \( Ax = b \) using factors from CPBCO or CPBFA.

- **CPTSL**
  - Solves a complex positive definite tridiagonal system of equations.
F4el. Linear Systems: Double-Precision Non-Symmetric Full

LINPACK

DGECO Factors a double precision matrix by Gaussian elimination and estimates the condition of the matrix.

DGEDI Computes the determinant and inverse of a matrix using the factors computed by DGECO or DGEFA.

DGEFA Factors a double precision matrix by Gaussian elimination.

DGESL Solves the double precision system $Ax = b$ or $A^T x = b$ using the factors computed by DGECO or DGEFA.

DTRCO Estimates the condition of a double precision triangular matrix.

DTRDI Computes the determinant and/or the inverse of an upper or lower double precision triangular matrix.

DTRSL Solves systems $Tx = b$ or $T^T x = b$ where $T$ is a triangular matrix of order $n$. Here $T^T$ denotes the transpose of the matrix $T$.

PORT

DLINEQ Double precision version of LINEQ.

SLATEC

DGECO Factors a double precision matrix by Gaussian elimination and estimates the condition of the matrix.

DGEDI Computes the determinant and inverse of a matrix using factors computed by DGECO or DGEFA.

DGEFA Factors a double precision matrix by Gaussian elimination.

DGEFS Solves a general double precision $n \times n$ system of linear equations.

DGESL Solves the double precision system $Ax = b$ or $A^T x = b$ using the factors computed by DGECO or DGEFA.

DTRCO Estimates the condition of a double precision triangular matrix.

DTRDI Computes the determinant and inverse of a double precision triangular matrix.

DTRSL Solves systems of the form $Tx = b$ or $T^T x = b$ where $T$ is a triangular double precision matrix of order $N$. 

---

Linear Algebra 184
### F4e2. Linear Systems: Double-Precision Non-Symmetric Banded

#### LINPACK

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGBCO</td>
<td>Factors a double precision band matrix by Gaussian elimination and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>DGBDI</td>
<td>Computes the determinant of a band matrix using the factors computed by DGBCO or DGBFA. If the inverse is needed, use DGBSL ( n ) times.</td>
</tr>
<tr>
<td>DGBFA</td>
<td>Factors a double precision band matrix by elimination.</td>
</tr>
<tr>
<td>DGBSL</td>
<td>Solves the double precision band system ( Ax = b ) or ( A^T x = b ) using the factors computed by DGBCO or DGBFA.</td>
</tr>
<tr>
<td>DGTSI</td>
<td>Given a general tridiagonal matrix and a right hand side will find the solution.</td>
</tr>
</tbody>
</table>

#### SLATEC

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGBCO</td>
<td>Factors a double precision band matrix by Gaussian elimination and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>DGBDI</td>
<td>Computes the determinant of a double precision band matrix using factors computed by DGBCO or DGBFA. Use DGBSL ( n ) times for the inverse.</td>
</tr>
<tr>
<td>DGBFA</td>
<td>Factors a double precision band matrix by elimination.</td>
</tr>
<tr>
<td>DGBSL</td>
<td>Solves the double precision band system ( Ax = b ) or ( A^T x = b ) using the factors computed by DGBCO or DGBFA.</td>
</tr>
<tr>
<td>DGTSI</td>
<td>Solves the system ( Tx = b ) where ( T ) is a general tridiagonal matrix. Double precision version.</td>
</tr>
<tr>
<td>DNBCO</td>
<td>Factors a double precision band matrix by Gaussian elimination and estimates the condition of the matrix.</td>
</tr>
<tr>
<td>DNBDO</td>
<td>Computes the determinant of a band matrix using the factors computed by DNBCO or DNBFA.</td>
</tr>
<tr>
<td>DNBFA</td>
<td>Factors a double precision band matrix by elimination.</td>
</tr>
<tr>
<td>DNBFS</td>
<td>Solves a general nonsymmetric banded double precision ( n \times n ) system of linear equations.</td>
</tr>
<tr>
<td>DNBSL</td>
<td>Solves a double precision band system using the factors computed by DNBCO or DNBFA.</td>
</tr>
</tbody>
</table>
F4fla. Linear Systems: Double-Precision Symmetric Full Indefinite

**LINPACK**

- **DSICO**: Factors a double precision symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
- **DSIDI**: Computes the determinant, inertia and inverse of a double precision symmetric matrix using the factors from DSIFA.
- **DSIFA**: Factors a double precision symmetric matrix by elimination with symmetric pivoting.
- **DSISL**: Solves the double precision symmetric system $Ax = b$ using the factors computed by DSIFA.
- **DSPCO**: Factors a double precision symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.
- **DSPDI**: Computes the determinant, inertia and inverse of a double precision symmetric matrix using factors from DSPFA, where the matrix is stored in packed form.
- **DSPFA**: Factors a double precision symmetric matrix stored in packed form by elimination with symmetric pivoting.
- **DSPSL**: Solves the double precision symmetric system $Ax = b$ using the factors computed by DSPFA.

**SLATEC**

- **DSICO**: Factors a double precision symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.
- **DSIDI**: Computes the determinant, inertia and inverse of a double precision symmetric matrix using the factors from DSIFA.
- **DSIFA**: Factors a double precision symmetric matrix by elimination with symmetric pivoting.
- **DSISL**: Solves the double precision symmetric system $Ax = b$ using the factors computed by DSIFA.
- **DSPCO**: Factors a double precision symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.
- **DSPDI**: Computes the determinant, inertia and inverse of a double precision symmetric matrix using the factors from DSPFA, where the matrix is stored in packed form. Double precision version.
- **DSPFA**: Factors a double precision symmetric matrix stored in packed form by elimination with symmetric pivoting.
- **DSPSL**: Solves the double precision symmetric system $Ax = b$ using the factors computed by DSPFA.
F4f1b. Linear Systems: Double-Precision Symmetric Full Positive-Definite

LINPACK

**DCHDC** Computes Cholesky factorization of a double precision positive definite matrix.

**DPOCO** Factors a double precision symmetric positive definite matrix and estimates the condition of the matrix.

**DPODI** Computes the determinant and inverse of a certain double precision, symmetric, positive definite matrix (see below) using the factors computed by DPOCO, DPOFA or DQRDC.

**DPOFA** Factors a double precision symmetric positive definite matrix.

**DPOFS** DPOFS solves a positive definite symmetric double precision $n \times n$ system of linear equations.

**DPOSL** Solves the double precision symmetric positive definite system $Ax = b$ using the factors computed by DPOCO or DPOFA.

**DPPCO** Factors a double precision symmetric positive definite matrix stored in packed form and estimates the condition of the matrix.

**DPPDI** Computes the determinant and inverse of a double precision symmetric positive definite matrix using the factors computed by DPPCO or DPPFA.

**DPPFA** Factors a double precision symmetric positive definite matrix stored in packed form.

**DPPSL** Solves the double precision symmetric positive definite system $Ax = b$ using the factors computed by DPPCO or DPPFA.

SLATEC

**DCHDC** Computes the Cholesky decomposition of a positive definite double precision matrix. A pivoting option allows the user to estimate the condition of a positive definite matrix or determine the rank of a positive semidefinite matrix.

**DPOCO** Factors a double precision symmetric positive definite matrix and estimates the condition of the matrix.

**DPODI** Computes the determinant and inverse of a certain double precision symmetric positive definite matrix using the factors computed by DPOCO, DPOFA, or DQRDC.

**DPOFA** Factors a double precision symmetric positive definite matrix.

**DPOFS** DPOFS solves a positive definite symmetric double precision $n \times n$ system of linear equations.

**DPOSL** Solves the double precision symmetric positive definite system $Ax = b$ using the factors computed by DPOCO or DPOFA.

**DPPCO** Factors a double precision symmetric positive definite matrix stored in packed form and estimates the condition of the matrix.

**DPPDI** Computes the determinant and inverse of a double precision symmetric positive definite matrix using factors from DPPCO or DPPFA.

**DPPFA** Factors a double precision symmetric positive definite matrix (packed form).
### Linear Systems: Double-Precision Symmetric Banded

**DPPSL** Solves the double precision symmetric positive definite system \( Ax = b \).

#### LINPACK

- **DPBCO** Factors a double precision symmetric positive definite matrix stored in band form and estimates the condition of the matrix.
- **DPBDI** Computes the determinant of a double precision symmetric positive definite band matrix using the factors computed by DPBCO or DPBFA. If the inverse is needed, use DPBSL \( n \) times.
- **DPBFA** Factors a double precision symmetric positive definite matrix stored in band form.
- **DPBSL** Solves the double precision symmetric positive definite band system \( Ax = b \) using the factors computed by DPBCO or DPBFA.
- **DPTSL** Given a positive definite tridiagonal matrix and a right hand side will find the solution.

#### SLATEC

- **DPBCO** Factors a double precision symmetric positive definite matrix stored in band form and estimates the condition of the matrix.
- **DPBDI** Computes the determinant of a double precision symmetric positive definite band matrix using factors of DPBCO or DPBFA. Use DPBSL \( n \) times for inverse.
- **DPBFA** Factors a double precision symmetric positive definite matrix stored in band form.
- **DPBSL** Solves the double precision symmetric positive definite band system \( Ax = b \) using the factors computed by DPBCO or DPBFA.
- **DPTSL** Solves the system \( Tx = b \) where \( T \) is positive definite symmetric tridiagonal. Double precision version.
F5. QR (Orthogonal) Decomposition

**LINPACK**

CQRDC  Uses Householder transformations to compute the QR factorization of an $n \times p$ complex matrix. Column pivoting based on the 2-norms of the reduced columns may be performed at the users option.

CQRSL  Applies the output of the subroutine CQRDC to compute projections and least squares solutions.

DQRDC  Uses Householder transformations to compute the QR factorization of an $n$ by $p$ matrix $X$. Column pivoting based on the 2-norms of the reduced columns may be performed at the users option.

DQRSL  Applies the output of the subroutine DQRDC to compute projections and least squares solutions.

SQRDC  Uses Householder transformations to compute the QR factorization of an $n \times p$ matrix $X$. Column pivoting based on the 2-norms of the reduced columns may be performed at the users option.

SQRSL  Applies the output of the subroutine SQRDC to compute projections and least squares solutions.

**NAG**

F02WDF  Returns the Householder QR factorisation of a real rectangular $m \times n$ ($m \geq n$) matrix $A$ and, either if $A$ is not of full rank or if requested, part or all of the singular value decomposition of $A$.

**SCILIB**

CQRDC  Cray 1 optimized version of the LINPACK routine that uses Householder transformations to compute the QR factorization of an $n \times p$ complex matrix. Column pivoting based on the 2-norms of the reduced columns may be performed at the users option.

CQRSL  Cray 1 optimized version of the LINPACK routine that applies the output of the subroutine CQRDC to compute projections and least squares solutions.

SQRDC  Cray 1 optimized version of the LINPACK routine; uses Householder transformations to compute the QR factorization of an $n \times p$ matrix $X$. Column pivoting based on the 2-norms of the reduced columns may be performed at the users option.

SQRSL  Cray 1 optimized version of the LINPACK routine; applies the output of the subroutine SQRDC to compute projections and least squares solutions.
CQRDC: Uses Householder transformations to compute the QR factorization of an \( n \times p \) matrix \( X \). Column pivoting is optional.

CQRSL: Applies the output of CQRDC to compute coordinate transformations, projections, and least squares solutions.

DQRDC: Uses Householder transformations to compute the QR factorization of \( n \times p \) matrix \( X \). Column pivoting is optional. Double precision version.

DQRSL: Applies the output of DQRDC to compute coordinate transformations, projections, and least squares solutions. Double precision version.

SQRDC: Uses Householder transformations to compute the QR factorization of an \( n \times p \) matrix \( X \). Column pivoting is a user's option.

SQRSL: Applies the output of SQRDC to compute coordinate transformations, projections, and least squares solutions.

---

F6. Singular Value Decomposition, Rank Determination, Generalized Inverse

EISPACK

MINFIT: Determines, towards the solution of the linear system \( Ax = b \), the singular value decomposition \( A = U S V^T \) of a real \( m \times n \) rectangular matrix, forming \( U^T b \) rather than \( U \). Householder bidiagonalization and a variant of the QR algorithm are used.

SVD: Determines the singular value decomposition \( A = U S V^T \) of a real \( m \times n \) rectangular matrix. Householder bidiagonalization and a variant of the QR algorithm are used.

EISPKD

MINFIT: Determines, towards the solution of the linear system \( Ax = b \), the singular value decomposition \( A = U S V^T \) of a double precision real \( m \times n \) rectangular matrix, forming \( U^T b \) rather than \( U \). Householder bidiagonalization and a variant of the QR algorithm are used.

SVD: Determines the singular value decomposition \( A = U S V^T \) of a double precision real \( m \times n \) rectangular matrix. Householder bidiagonalization and a variant of the QR algorithm are used.


**IMSL**

- **LGINF** Generalized inverse of a real matrix.
- **LSVDB** Singular value decomposition of a bidiagonal matrix.
- **LSVDF** Singular value decomposition of a real matrix.

**LINPACK**

- **CSVDC** Reduces a complex $n \times p$ matrix $X$ by unitary transformations $U$ and $V$ to diagonal form. The diagonal elements $s_i$ are the singular values of $X$. The columns of $U$ are the corresponding left singular vectors, and the columns of $V$ the right singular vectors.
- **DSVDC** Reduces a double precision $n \times p$ matrix $X$ by orthogonal transformations $U$ and $V$ to diagonal form. The diagonal elements $s_i$ are the singular values of $X$. The columns of $U$ are the corresponding left singular vectors, and the columns of $V$ the right singular vectors.
- **SSVDC** Reduces a real $n \times p$ matrix $X$ by orthogonal transformations $U$ and $V$ to diagonal form. The diagonal elements $s_i$ are the singular values of $X$. The columns of $U$ are the corresponding left singular vectors, and the columns of $V$ the right singular vectors.

**NAG**

- **F01BLF** Calculates the rank and pseudo-inverse of an $m \times n$ real matrix $A$, $m \geq n$, $\text{rank}(A) \leq n$, using Householder's factorisation.
- **F01BUF** Decomposes a symmetric positive definite band matrix into the form $ULDL^T U^T$ where $U$ is a unit upper triangular matrix, $L$ is unit lower triangular and $D$ is diagonal. It is specifically designed to precede F01BVF.
- **F01QAF** QU-factorization of an $m \times n$ real matrix $A$ ($m \geq n$).
- **F01QBF** QU-factorization of an $m \times n$ real matrix $A$ ($m \leq n$).
- **F02SZF** Calculates the singular value decomposition of a real $n \times n$ upper bidiagonal matrix $A$.
- **F02WAF** Finds the singular values and the right-hand singular vectors (principal components) of a real rectangular $m \times n$ matrix $A$, where $m \geq n$.
- **F02WBF** Finds the singular values and right-hand singular vectors (principal components) of a real rectangular $m \times n$ matrix $A$, where $m \leq n$.
- **F02WCF** Computes the singular values and left- and right-hand singular vectors of a real rectangular $m \times n$ matrix $A$, $A = QDP^T$, where $Q^T Q = P^T P = I(k)$, $k = \min(m, n)$ and $D = \text{diag}(sv_1, sv_2, \ldots, sv_k)$ with $sv_1 \geq sv_2 \geq \cdots \geq sv_k \geq 0$.
- **F02WDF** Returns the Householder QU factorisation of a real rectangular $m \times n$ ($m \geq n$) matrix $A$ and, either if $A$ is not of full rank or if requested, part or all of the singular value decomposition of $A$.
NCARLB

**LSQSLV**
Determines the elements of the singular value decomposition of a given matrix for use in the computation of the least squares solution of an overdetermined linear system.

**SVDSLV**
Determines singular values and the least squares solution of a large overdetermined linear system. Requires only one row of the coefficient matrix at a time, thus allowing the solution of systems too large to fit in core.

**SCILIB**

**CSVDC**
Cray 1 optimized version of the LINPACK routine that reduces a complex \( n \times p \) matrix \( X \) by unitary transformations \( U \) and \( V \) to diagonal form. The diagonal elements \( s_i \) are the singular values of \( X \). The columns of \( U \) are the corresponding left singular vectors, and the columns of \( V \) the right singular vectors.

**MINFIT**
Cray 1 optimized version of the EISPACK routine; determines, towards the solution of the linear system \( Ax = b \), the singular value decomposition \( A = UsV^T \) of a real \( m \times n \) rectangular matrix, forming \( U^T b \) rather than \( U \). Householder bidiagonalization and a variant of the QR algorithm are used.

**SSVDC**
Cray 1 optimized version of the LINPACK routine; a subroutine to reduce a real \( n \times p \) matrix \( X \) by orthogonal transformations \( U \) and \( V \) to diagonal form. The diagonal elements \( s_i \) are the singular values of \( X \). The columns of \( U \) are the corresponding left singular vectors, and the columns of \( V \) the right singular vectors.

**SVD**
Cray 1 optimized version of the EISPACK routine; determines the singular value decomposition \( A = UsV^T \) of a real \( m \times n \) rectangular matrix. Householder bidiagonalization and a variant of the QR algorithm are used.

**SLATEC**

**CSVDC**
Performs the singular value decomposition of a complex \( n \times p \) matrix.

**DSVDC**
Performs the singular value decomposition of a double precision \( n \times p \) matrix.

**MINFIT**
Computes singular value decomposition of rectangular matrix and solve related linear least squares problem.

**SVD**
Determines the singular value decomposition \( A = UsV^T \) of a real \( m \times n \) rectangular matrix. Householder bidiagonalization and a variant of the QR algorithm are used.

**SSVDC**
Performs the singular value decomposition of a real \( n \times p \) matrix.

**ULSIA**
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.
F7. Updating Matrix Decompositions

**LINPACK**

CCHDD  Downdates the Cholesky factorization of a complex positive definite matrix.

CCHEX  Updates the Cholesky factorization of a complex positive definite matrix after symmetric permutations of rows and columns.

CCHUD  Updates an augmented Cholesky factorization of a triangular part of an augmented QR decomposition of a complex positive definite matrix.

DCHDD  Downdates the Cholesky factorization of a double precision positive definite matrix.

DCHEX  Updates the Cholesky factorization of a double precision positive definite matrix after symmetric permutations of rows and columns.

DCHUD  Updates an augmented Cholesky factorization of a triangular part of an augmented QR decomposition of a double precision positive definite matrix.

SCHDD  Downdates the Cholesky factorization of a real positive definite matrix.

SCHEX  Updates the Cholesky factorization of a real positive definite matrix after symmetric permutations of rows and columns.

SCHUD  Updates an augmented Cholesky factorization of a triangular part of an augmented QR decomposition of a real positive definite matrix.

**NAG**

F05AAF  Applies the Schmidt orthogonalisation process to $N$ vectors in $M$ dimensional space, $N \leq M$.

**SCILIB**

CCHDD  Cray 1 optimized version of the LINPACK routine that downdates the Cholesky factorization of a complex positive definite matrix.

CCHEX  Cray 1 optimized version of the LINPACK routine that updates the Cholesky factorization of a complex positive definite matrix after symmetric permutations of rows and columns.

CCHUD  Cray 1 optimized version of the LINPACK routine that updates an augmented Cholesky factorization of a triangular part of an augmented QR decomposition of a complex positive definite matrix.

SCHDD  Cray 1 optimized version of the LINPACK routine; downdates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.

SCHEX  Cray 1 optimized version of the LINPACK routine; updates the Cholesky factorization of a positive definite matrix under diagonal permutations.

SCHUD  Cray 1 optimized version of the LINPACK routine; updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.
SLATEC

**CCHDD**
Downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.

**CCHEX**
Updates the Cholesky factorization $A = R^H R$ of a positive definite matrix $A$ of order $P$ under diagonal permutations of the form $U^R E = R'$, where $E$ is a permutation matrix.

**CCHUD**
Updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.

**DCHDD**
Downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition of positive definite double precision matrix.

**DCHEX**
Updates the Cholesky factorization $A = R^T R$ of a positive definite double precision matrix $A$ of order $P$ under diagonal permutations of the form $E^T A E$ where $E$ is a permutation matrix.

**DCHUD**
Updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition. Double precision version.

**SCHDD**
Downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition.

**SCHEX**
Updates the Cholesky factorization $A = R^T R$ of a positive definite matrix $A$ of order $P$ under diagonal permutations of the form $E^T A E$ where $E$ is a permutation matrix.

**SCHUD**
Updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.
G1. Fast Fourier Transforms, Sine and Cosine Transforms

IMSL

FFT2C Compute the fast Fourier transform of a complex valued sequence of length equal to a power two.

FFT3D Compute the fast fourier transform of a complex valued 1, 2 or 3 dimensional array.

FFTCC Compute the fast Fourier transform of a complex valued sequence.

FFTRC Compute the fast Fourier transform of a real valued sequence.

FFTSC Compute the sine and cosine transforms of a real valued sequence.

FTFPS Fast Fourier transform estimates of power spectra and cross spectra of time series.

LOCLIB

CFFT99 Performs multiple fast Fourier transforms. This package will perform a number of simultaneous complex periodic Fourier transforms or corresponding inverse transforms. The length of the transforms must be a number greater than 1 that has no prime factors other than 2, 3, and 5. Multi-dimensional fast Fourier transforms are supported by successive calls to the routines in this package.

FFAFFS Performs an in-place finite discrete Fourier transform and Fourier synthesis on real data.

FFT99F Fortran source for FORTRAN version of Temperton's vectorized multiple FFT package. Performs a number of simultaneous real/half-complex periodic Fourier transforms or corresponding inverse transforms. The length of the transforms must be an even number greater than 4 with no prime factors greater than 5. This is an all-FORTRAN version of the faster LOCLIB package that is written in FORTRAN and CAL.

OLDFFTPK Very old, very slow version of FFTPACK. It is kept for the convenience of users who cannot change to FFTPACK, and eventually will be deleted.

NAG

C06ADF Calculates the finite Fourier transform of one variable of dimension NV within multivariable transform of N complex data values.

C06EAF Calculates the discrete Fourier transform of a sequence of N real data values. (No extra workspace required.)

C06EBF Calculates the discrete Fourier transform of a Hermitian sequence of N complex data values. (No extra workspace required.)
Calculates the discrete Fourier transform of a sequence of $N$ complex data values. (No extra workspace required.)

Calculates the discrete Fourier transform of a sequence of $N$ real data values (using a work array for extra speed).

Calculates the discrete Fourier transform of a Hermitian sequence of $N$ complex data values (using a work array for extra speed).

Calculates the discrete Fourier transform of a sequence of $N$ complex data values (using a work array for extra speed).

Computes the discrete Fourier transform of one variable in a multivariate sequence of complex data values.

Computes the multi-dimensional discrete Fourier transform of a multivariate sequence of complex data values.

Cray assembly language source for Temperton's vectorized multiple FFT package, which performs a number of simultaneous real/half-complex periodic Fourier transforms or corresponding inverse transforms. Non-proprietary; requires FFT99 package.

Computes fast Fourier transforms for data vectors of arbitrary length.

Portran source for Temperton's vectorized multiple FFT package. Performs a number of simultaneous real/half-complex periodic Fourier transforms or corresponding inverse transforms. The length of the transforms must be an even number with no prime factors greater than 5. Non-proprietary; requires CAL99 package.

Computes symmetric fast Fourier transforms and quarter wave transforms.

Computes fast Fourier transforms for data vectors whose length is a power of two.

Double precision version of FFT.

Double precision version of FFTC.

Double precision version of FFTCI.

Double precision version of FFTR.

Double precision version of FFTRI.

Uses a mixed radix fast Fourier transform method to find the discrete Fourier transform or its (unscaled) inverse of complex-valued data. FFT can be used for multivariate transforms. Singleton's algorithm is used.

Performs a mixed radix fast Fourier transform of complex data. FFTC has a simpler argument list than the more general PORT subroutine FFT.

Finds the inverse fast Fourier transform of complex data, given the Fourier coefficients in the frequency domain. FFTCI has a simpler argument list than the more general PORT subroutine FFT.
**NCAR Software Catalog**  

**FFTR**  
Performs a mixed radix fast Fourier transform to find the transform of $2N$ real data points. FFTR has a simpler argument list than the more general PORT subroutine FFT.

**FFTRI**  
Finds the inverse Fourier transform using Fourier coefficients presumed to arise from real data in the time domain. FFTRI has a simpler argument list than the more general PORT subroutine FFT.

**SCILIB**

**CFFT2**  
Applies a forward or reverse complex-to-complex Fourier transform on a vector of length $2^n$, where $3 \leq n \leq 16$. This routine has been optimized for the Cray 1.

**CRFFT2**  
Applies a forward or reverse complex-to-real Fourier transform on a symmetric complex vector of length $2^n$, where $3 \leq n \leq 16$. This routine has been optimized for the Cray 1.

**RCFFT2**  
Applies a forward or reverse real-to-complex Fourier transform on a vector of length $2^n$, where $3 \leq n \leq 16$. This routine has been optimized for the Cray 1.

**SLATEC**

**CFFTB**  
Unnormalized inverse of CFFTF.

**CFFTF**  
Forward transform of a complex, periodic sequence.

**CFFTI**  
Initialize for CFFTF and CFFTB.

**COSQB**  
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**  
A Simplified real, periodic, backward transform.

**EZFFTF**  
A simplified real, periodic, forward transform.

**EZFFTI**  
Initialize EZFFTF and EZFFTB.

**RFFTB**  
Backward transform of a real coefficient array.

**RFFTF**  
Forward transform of a real, periodic sequence.

**RFFTI**  
Initialize for RFFTF and RFFTB.

**SINQB**  
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.
G2. Convolutions

IMSL

VCONVO Vector convolution.

NAG

C06ACF Calculates the circular convolution of two real vectors of period $2^M$ by the fast Fourier transform. $M$ is a positive integer not less than 2.

C06EKF Calculates the circular convolution or correlation of two real vectors of period $N$. (No extra workspace required.)

C06FKF Calculates the circular convolution or correlation of two real vectors of period $N$ (using a work array for extra speed).

G3. Laplace Transforms

IMSL

FLINV Inverse Laplace transform of a user supplied complex function.

G4. Hilbert Transforms

SLATEC

DQAWC Calculates an approximation result to a Cauchy principal value $I = \int_{a}^{b} f(x)w(x)dx$ ($w(x) = \frac{1}{x - c}, c \neq a, c \neq b$), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \max(e_{\text{abs}}, e_{\text{rel}} |I|)$. Double precision version.

DQAWCE Calculates an approximation result to a Cauchy principal value $I = \int_{a}^{b} f(x)w(x)dx$ ($w(x) = \frac{1}{x - c}, c \neq a, c \neq b$), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \max(e_{\text{abs}}, e_{\text{rel}} |I|)$. Provides more information than DQAWC. Double precision version.

DQC25C Computes $I = \int_{a}^{b} f(x)w(x)dx$ with error estimate, where $w(x) = \frac{1}{x - c}$. Double precision version.
QAWC   Calculates an approximation result to a Cauchy principal value
       \[ I = \int_a^b f(x)w(x)dx \quad (w(x) = \frac{1}{x-c}, c \neq a, c \neq b) \], attempting to satisfy the following accuracy criterion:
       \[ |I - \text{result}| \leq \max(\varepsilon_{abs}, \varepsilon_{rel} |I|). \]

QAWCE  Calculates an approximation result to a Cauchy principal value
       \[ I = \int_a^b f(x)w(x)dx \quad (w(x) = \frac{1}{x-c}, c \neq a, c \neq b) \], attempting to satisfy the following criterion:
       \[ |I - \text{result}| \leq \max(\varepsilon_{abs}, \varepsilon_{rel} |I|). \] Provides more information than QAWC.

QC25C  Computes \[ I = \int_a^b f(x)w(x)dx \] with error estimate, where \( w(x) = \frac{1}{x-c} \)
H1a. Numerical Differentiation: Function Available only on Grid

**FITPACK**

**CURV2D**
Determines function value, first and second derivatives of a curve at a given point using a spline under tension. The subroutine CURV1 should be called earlier to determine certain necessary parameters.

**CURVD**
Differentiates a curve at a given point using a spline under tension. The subroutine CURV1 should be called earlier to determine certain necessary parameters.

**CURVPD**
Differentiates a curve at a given point using a periodic spline under tension. The subroutine CURVP1 should be called earlier to determine certain necessary parameters.

**KURVD**
Performs the mapping of points in the interval (0,1) onto a curve in the plane. It also returns the first and second derivatives of the component functions. The subroutine KURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation both of whose components are splines under tension and functions of the polygonal arclength parameter.

**KURVPD**
Performs the mapping of points in the interval (0,1) onto a closed curve in the plane. It also returns the first and second derivatives of the component functions. The subroutine KURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation both of whose components are periodic splines under tension and functions of the polygonal arclength parameter.

**QURVD**
Performs the mapping of points in the interval (0,1) onto a curve in space. It also returns the first and second derivatives of the component functions. The subroutine QURV1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation all of whose components are splines under tension and functions of the polygonal arclength parameter.

**QURVPD**
Performs the mapping of points in the interval (0,1) onto a closed curve in three-dimensional space. It also returns the first and second derivatives of the component functions. The subroutine QURVP1 should be called earlier to determine certain necessary parameters. The resulting curve has a parametric representation all of whose components are periodic splines under tension and functions of the polygonal arclength parameter.

**SURFBD**
Evaluates the function value, the two first partial derivatives and the three second partial derivatives of a tensor product spline under tension in two variables. The subroutine SURFBD should be called earlier to determine certain necessary parameters.

**SURFD**
Evaluates the function value, the two first partial derivatives and the three second partial derivatives of a tensor product spline under tension in two variables. The subroutine SURFD should be called earlier to determine certain necessary parameters.
necessary parameters.

**VAL3BD**
Evaluates the function value, the three first partial derivatives and the six second partial derivatives of a tensor product spline under tension in three variables. The subroutine VAL3B1 should be called earlier to determine certain necessary parameters.

**VAL3HD**
Evaluates the function value, the three first partial derivatives and the six second partial derivatives of a tensor product spline under tension in three variables.

**ZURF2D**
Performs the mapping of points in the square \((0,1) \times (0,1)\) onto a surface and derivatives with respect to two parameters for use in calculating normal vectors, tangent planes, curvatures, etc. The subroutine ZURF2N should be called earlier to determine certain necessary parameters. The resulting surface has a parameteric representation all three of whose components are splines under tension and functions of the two grid parameters.

**IMSL**

**DBCEVL**
Bicubic spline mixed partial derivative evaluator.

**DCSEVU**
Cubic spline first and second derivative evaluator.

**LOCLIB**

**FINDIF**
Given a function defined on a regularly-spaced mesh, this subroutine computes a closed form finite difference formula for the \(n\)-th derivative of the function at a given mesh point.

**FINPDF**
Finds a finite difference formula in integer closed form for the mixed partial derivative of a function given on an \(n\)-dimensional non-uniform product mesh at a specified point.

**NAG**

**E02BCF**
Evaluates a cubic spline and its first three derivatives from its B-spline representation.

**NCARLB**

**CUBSPL**
Performs one- and two-dimensional cubic spline interpolation for values and first and second derivatives.

**PORT**

**BSPL1**
Evaluates, at a given set of points in a specified mesh interval, basis splines together with selected orders of derivatives.

**BSPLD**
Evaluates basis splines and their derivatives at a given set of points in a specified mesh interval.

**CSPDI**
Cubic spline differentiation. Finds a numerical approximation to the first derivative at requested points in given input data using a spline approximation. The spline function used is an interpolatory fit going through all the input data points.
NCAR Software Catalog

DBSPL1 Double precision version of BSPL1.
DBSPLD Double precision version of BSPLD.
DCSPDI Double precision version of CSPDI.
DL2SFH Double precision version of L2SFH.
DSPLN1 Double precision version of SPLN1.
DSPLND Double precision version of SPLND.
L2SFH Fits B-splines to the derivatives of a function. Obtains a B-spline approximation to a function and its derivatives through a specified order, by minimizing a functional involving least-squares integrals on a given mesh. The minimization determines coefficients in an expansion of the function in terms of B-splines of a specified order.
SPLND Spline and derivatives evaluation. Evaluates, at a given set of points, a function described by a (previously determined) expansion in terms of B-splines. Derivatives of the sum (evaluated at the same given points) through a specified order are also returned. If many calls with a small number of points of evaluation to SPLND are anticipated, it would be cheaper to use SPLN2, since SPLN2 saves intermediate results from one call to the next.

SLATEC

BSPDR Uses the $B$-representation to construct a divided difference table preparatory to a (right) derivative calculation in BSPEV.
BSPEV Calculates the value of the spline and its derivatives at $x$ from the $B$-representation.
BSPVD Calculates the value and all derivatives of order less than $nderiv$ of all basis functions which do not vanish at $x$.
BVALU Evaluates the $B$-representation of a $B$-spline at $x$ for the function value or any of its derivatives.
CHFDV Evaluates a cubic polynomial given in Hermite form and its first derivative at an array of points.
DBSPDR Uses the $B$-representation to construct a divided difference table ad preparatory to a (right) derivative calculation in BSPEV. Double precision version.
DBSPEV Calculates the value of the spline and its derivatives at $x$ from the $B$-representation. Double precision version.
DBSPVD Calculates the value and all derivatives of order less than $nderiv$ of all basis functions which do not vanish at $x$. Double precision version.
DBVALU Evaluates the $B$-representation of a $B$-spline at $x$ for the function value or any of its derivatives. Double precision version.
DPPVAL Calculates (at $x$) the value of the ideriv-th derivative of the $B$-spline from the piecewise polynomial representation. Double precision version.
PCHFD Evaluates the piecewise cubic interpolant at an array of points and has the capability to evaluate internal derivatives of the interpolant.
PPVAL Calculates (at $x$) the value of the ideriv-th derivative of the $B$-spline from the piecewise polynomial representation.

Differentiation, Integration
H1b. Numerical Differentiation: User-Defined Function

**IMSL**

**DRVTE** Calculate first, second or third derivative of a user supplied function.

**MINPACK**

**CHKDER** Checks a Jacobian matrix of partial derivatives of a multivariate function for consistency with the function values. This should be used to check user-coded Jacobians, since providing a Jacobian matrix can be an error-prone task.

**NAG**

**D04AAF** Calculates a set of derivatives (up to order 14) of a function of one real variable at a point, together with a corresponding set of error estimates, using an extension of the Neville algorithm.

**E02AHF** Determines the coefficients in the Chebyshev-series representation of the derivative of a polynomial given in Chebyshev-series form.

**E04HBF** Computes a sensible set of finite-difference intervals for input to a quasi-Newton minimization routine which does not require derivatives.

**E04HCF** Checks that a user-supplied routine for evaluating an objective function and its first derivative produces derivative values which are consistent with the function values calculated.

**E04HDF** Checks that a user-supplied routine for calculating second derivatives of an objective function is consistent with a user-supplied routine for calculating the corresponding first derivatives.

**E04YAF** Checks that a user-supplied routine for evaluating a vector of functions and the matrix of their first derivatives produces derivative values which are consistent with the function values calculated.

**E04YBF** Checks that a user-supplied routine for evaluating the second derivative term of the Hessian matrix of a sum of squares is consistent with a user-supplied routine for calculating the corresponding first derivatives.

**E04ZAF** Checks that user-supplied routines for evaluating an objective function, constraint functions and their first derivative produce derivative values which are consistent with the function and constraint values calculated.

**E04ZBF** Checks that user-supplied routines for calculating second derivatives of an objective function and of constraint functions are consistent with user-supplied routines for calculating the corresponding first derivatives.

**E04ZCF** Checks that user-supplied routines for evaluating an objective function, constraint functions and their first derivatives produce derivative values which are consistent with the function and constraint values calculated.
**SLATEC**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKDER</td>
<td>Checks the gradients of $m$ nonlinear functions in $n$ variables, evaluated at a point $x$, for consistency with the functions themselves.</td>
</tr>
<tr>
<td>DCKDER</td>
<td>Checks the gradients of $m$ nonlinear functions in $n$ variables, evaluated at a point $x$, for consistency with the functions themselves. Double precision version.</td>
</tr>
</tbody>
</table>

---

**H2a1. One-Dimensional Integration: Integrand Defined on a Grid**

**FITPACK**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARKUR</td>
<td>Integrates a curve in the plane specified by a spline under tension between two given limits. The subroutine KURVI should be called earlier to determine necessary parameters.</td>
</tr>
<tr>
<td>ARKURP</td>
<td>Integrates a closed curve in the plane specified by a spline under tension between two given limits. The subroutine KURVP1 should be called earlier to determine necessary parameters.</td>
</tr>
<tr>
<td>ARQUR</td>
<td>Integrates a curve in three dimensions specified by a spline under tension between two given limits. The subroutine QURVI should be called earlier to determine necessary parameters.</td>
</tr>
<tr>
<td>ARQURP</td>
<td>Integrates a closed curve in three dimensions specified by a spline under tension between two given limits. The subroutine QURVP1 should be called earlier to determine necessary parameters.</td>
</tr>
<tr>
<td>CURVI</td>
<td>Integrates a curve specified by a spline under tension between two given limits. The subroutine CURV1 should be called earlier to determine certain necessary parameters.</td>
</tr>
<tr>
<td>CURVPI</td>
<td>Integrates a curve specified by a periodic spline under tension between two given limits. The subroutine CURVP1 should be called earlier to determine certain necessary parameters.</td>
</tr>
</tbody>
</table>

**IMSL**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCSQDU</td>
<td>Cubic spline quadrature.</td>
</tr>
</tbody>
</table>

**LOCLIB**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIORDQ</td>
<td>Computes the integral of a function of one variable using the trapezoidal rule with Richardson extrapolation for an improved estimate.</td>
</tr>
</tbody>
</table>
**NAG**

**D01GAF**
Integrates a function which is specified numerically at four or more points, over the whole of its specified range, using third-order finite-difference formulae with error estimates, according to a method due to Gill and Miller.

**E02BDF**
Computes the definite integral of a cubic spline from its B-spline representation.

**NCARLB**

**SIMPSN**
Integrates a function using Simpson’s rule if given at equally spaced abscissas or using interpolatory quadratics if given at unequally spaced abscissas.

**PORT**

**BSPLI**
Obtains the integrals of basis splines, from the left-most mesh point to a specified set of points.

**CSPQU**
Cubic spline quadrature. Finds the integral of a function defined by pairs, \((x, y)\), of input points. The \(x\)'s do not have to be equally spaced. The spline function used is an interpolatory fit going through all the input data points.

**DBSPLI**
Double precision version of BSPLI.

**DCSPQU**
Double precision version of CSPQU.

**DSPLNI**
Double precision version of SPLNI.

**SPLNI**
Spline integration. Integrates a function described by a (previously determined) expansion in terms of B-splines. Several integrations can be performed in one call -- for each integral the lower limit is the left-hand mesh point and the upper limit is another point specified by the user.

**SLATEC**

**AVINT**
Integrate a function tabulated at arbitrarily spaced abscissas using overlapping parabolas.

**BFQAD**
Computes the integral on \((x_1, x_2)\) of a product of a function \(F\) and the \(id^{th}\) derivative of a \(k^{th}\) order B-spline (B-representation).

**BSQAD**
Computes the integral on \((x_1, x_2)\) of a \(k^{th}\) order B-spline using the B-representation.

**DAVINT**
Integrate a function tabulated at arbitrarily spaced abscissas using overlapping parabolas. Double precision version.

**DBFQAD**
Computes the integral on \((x_1, x_2)\) of a product of a function \(F\) and the \(id^{th}\) derivative of a \(k^{th}\) order B-spline (B-representation). Double precision version.

**DBSQAD**
Computes the integral on \((x_1, x_2)\) of a \(k^{th}\) order B-spline using the B-representation. Double precision version.

**DPFQAD**
Computes the integral on \((x_1, x_2)\) of a product of a function \(F\) and the \(id^{th}\) derivative of a B-spline, (piecewise polynomial representation).

**DPPQAD**
Computes the integral on \((x_1, x_2)\) of a \(k^{th}\) order B-spline using the piecewise polynomial representation.

**PFQAD**
Computes the integral on \((x_1, x_2)\) of a product of a function \(F\) and the \(id^{th}\) derivative of a B-spline, (piecewise polynomial representation).
PPQAD

Computes the integral on \((x_1, x_2)\) of a \(k^{th}\) order \(B\)-spline using the piecewise polynomial representation.

H2a2a. One-Dimensional Integration: Finite Interval, Nonautomatic, General Integrand

**NAG**

**D01BAF**

Computes an estimate of the definite integral of a function of known analytical form, using a Gaussian quadrature formula with a specified number of abscissae. Formulae are provided for a finite interval (Gauss-Legendre), a semi-infinite interval (Gauss-Laguerre, Gauss-Rational), and an infinite interval (Gauss-Hermite).

**D01BDF**

Calculates an approximation to the integral of a function over a finite interval \((A, B)\). It is non-adaptive and as such is recommended for the integration of “smooth” functions. These **exclude** integrands with singularities, derivative singularities or high peaks on \((A, B)\), or which oscillate too strongly on \((A, B)\).

**PORT**

**DGQ11**

Double precision version of GQM11.

**SLATEC**

**DQK15**

Computes \(I = \int_a^b f(x) \, dx\), with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 15-point Kronrod rule. Double precision version.

**DQK21**

Computes \(I = \int_a^b f(x) \, dx\), with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 21-point Kronrod rule. Double precision version.

**DQK31**

Computes \(I = \int_a^b f(x) \, dx\) with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 31-point Kronrod rule. Double precision version.

**DQK41**

Computes \(I = \int_a^b f(x) \, dx\), with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 41-point Kronrod rule. Double precision version.

**DQK51**

Computes \(I = \int_a^b f(x) \, dx\), with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 51-point Kronrod rule. Double precision version.

**DQK61**

Computes \(I = \int_a^b f(x) \, dx\), with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 61-point Kronrod rule. Double precision version.

**DQNG**

Calculates an approximation \(\text{result}\) to a given definite integral \(I = \int_a^b f(x) \, dx\), attempting to satisfy the following accuracy criterion: 
\[ |I - \text{result}| \leq \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}} |I|) \]. Non-adaptive integrator intended for smooth functions. Double precision version.

**QK15**

Computes \(I = \int_a^b f(x) \, dx\), with error estimate and \(J = \int_a^b |f(x)| \, dx\) using a 15-point Kronrod rule.

Differentiation, Integration 186
QK21 Computes \( I = \int_a^b f(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)| \, dx \) using a 21-point Kronrod rule.

QK31 Computes \( I = \int_a^b f(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)| \, dx \) using a 31-point Kronrod rule.

QK41 Computes \( I = \int_a^b f(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)| \, dx \) using a 41-point Kronrod rule.

QK51 Computes \( I = \int_a^b f(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)| \, dx \) using a 51-point Kronrod rule.

QK61 Computes \( I = \int_a^b f(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)| \, dx \) using a 61-point Kronrod rule.

QNG Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) \, dx \), attempting to satisfy the following accuracy criterion:

\[
| I - \text{result} | \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} \cdot |I|).
\]

Non-adaptive integrator intended for smooth functions.

H2a2b. One-Dimensional Integration: Finite Interval, Nonautomatic, Weighted Integrand

\textit{SLATEC}

DQC25C Computes \( I = \int_a^b f(x)w(x) \, dx \) with error estimate, where \( w(x) = \frac{1}{x-c} \). Double precision version.

DQC25F Computes the integral \( I = \int_a^b f(x)w(x) \, dx \) where \( w(x) = \cos(\omega x) \) or \( w(x) = \sin(\omega x) \) and \( J = \int_a^b |f(x)w(x)| \, dx \). For small value of \( \omega \) or small intervals \((a, b)\) the 15-point Gauss-Kronrod Rule is used. Otherwise a generalized Clenshaw-Curtis method is used. Double precision version.

DQC25S Computes \( I = \int_a^b f(x)w(x) \, dx \), with error estimate, where the weight function \( w \) has a singular behavior of algebraico-logarithmic type at the points \( a \) and/or \( b \). \((bl, br)\) is a part of \((a, b)\). Double precision version.

DQK15W Computes \( I = \int_a^b f(x)w(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)w(x)| \, dx \) using a 15-point Kronrod rule. Double precision version.

QC25C Computes \( I = \int_a^b f(x)w(x) \, dx \) with error estimate, where \( w(x) = \frac{1}{x-c} \).

QC25F Computes the integral \( I = \int_a^b f(x)w(x) \, dx \) where \( w(x) = \cos(\omega x) \) or \( w(x) = \sin(\omega x) \) and \( J = \int_a^b |f(x)w(x)| \, dx \). For small value of \( \omega \) or small intervals \((a, b)\) the 15-point Gauss-Kronrod rule is used. Otherwise a generalized Clenshaw-Curtis method is used.

QC25S Computes \( I = \int_a^b f(x)w(x) \, dx \), with error estimate, where the weight function \( w \) has a singular behavior of algebraico-logarithmic type at the points \( a \) and/or \( b \). \((bl, br)\) is a part of \((a, b)\).
computes \( I = \int_a^b f(x)w(x) \, dx \), with error estimate and \( J = \int_a^b |f(x)w(x)| \, dx \) using a 15-point Kronrod rule.

### H2a3a1. One-Dimensional Integration: Finite Interval, Automatic, General Integrand

#### AMOSLIB

**GAUS8** Integrates real functions of one variable over finite intervals.

#### IMSL

**DCADRE** Numerical integration of a function using cautious adaptive Romberg extrapolation.

#### NAG

**D01AHF** Computes a definite integral over a finite range to a specified relative accuracy using a method described by Patterson.

**D01AJF** Is a general-purpose integrator which calculates an approximation to the integral of a function \( F(x) \) over a finite interval \((A,B)\).

**D01AKF** Is an adaptive integrator, especially suited to oscillating, non-singular integrands, which calculates an approximation to the integral of a function \( F(x) \) over a finite interval \((A,B)\).

**D01ARF** Calculates definite and indefinite integrals over a finite interval to a specified accuracy.

#### NCARLB

**ADQUAD** Integrates a function over a finite interval within a specified relative error using adaptive subdivision of the interval to minimize the number of function evaluations required.

#### PORT

**DBQUAD** Double precision version of BQUAD.

**DODEQ** Double precision version of ODEQ.

**DQUAD** Double precision version of QUAD.

**DRQUAD** Double precision version of RQUAD.

**ODEQ** Integrates a set of functions over the same interval using the PORT differential equation subprogram ODES1. If the functions are not complicated and behave similarly over the integration interval, ODEQ will probably prove more efficient for three or more integrals than a quadrature routine which does integrals one at a time.
a time. This is particularly true when all the functions have a common piece which is expensive to evaluate.

**SLATEC**

DGAUS8 Integrates real functions of one variable over finite intervals using an adaptive 8-point Legendre-Gauss algorithm. DGAUS8 is intended primarily for high accuracy integration or integration of smooth functions. Double precision version.

DQAG Calculates an approximation result to a given definite integral \( I=\int_{a}^{b} f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
| I - \text{result} | \leq \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}} | I | )
\]. Double precision version.

DQAGE Calculates an approximation result to a given definite integral \( I=\int_{a}^{b} f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
| I - \text{result} | \leq \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}} | I | )
\]. Provides more information than DQAG. Double precision version.

DQNC79 Integrates a user defined function by a 7-point adaptive Newton-Cotes quadrature rule. Double precision version.

GAUS8 Integrates real functions of one variable over finite intervals using an adaptive 8-point Legendre-Gauss algorithm. GAUS8 is intended primarily for high accuracy integration or integration of smooth functions.

QAG Calculates an approximation result to a given definite integral \( I=\int_{a}^{b} f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
| I - \text{result} | \leq \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}} | I | )
\).

QAGE Calculates an approximation result to a given definite integral \( I=\int_{a}^{b} f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
| I - \text{result} | \leq \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}} | I | )
\]. Provides more information than QAG.

QNC79 Integrates a user defined function by a 7-point adaptive Newton-Cotes quadrature rule.

**H2a3a2. One-Dimensional Integration: Finite Interval, Automatic, General Integrand with Singularities or Discontinuities**

**NAG**

D01ALF Is a general purpose integrator which calculates an approximation to the integral of a function \( F(x) \) over a finite interval \((A,B)\), where the integrand may have local singular behaviour at a finite number of points within the integration interval.
PORT

BQUAD
Integrates a piecewise-smooth function. Should be used to integrate functions that have discontinuities in their derivatives. By putting in the x-values of the points of discontinuity, the user can make the integration process more economical than in the general case. Cautious adaptive Romberg extrapolation is used. Endpoint singularities are recognized and an automatic change of variable is made to handle them. Noisy integrands are integrated only to an accuracy commensurate with the noise.

QUAD
An adaptive quadrature routine for general use. In difficult cases, such as the presence of a singularity or noise, the routine attempts to obtain the best possible answer without wasting too much effort. Cautious adaptive Romberg extrapolation is used. The user must specify an absolute error criterion. Endpoint singularities are recognized and an automatic change of variable is made to handle them.

RQUAD
An adaptive quadrature routine for general use. In difficult cases, such as the presence of a singularity or noise, the routine attempts to obtain the best possible answer without wasting too much effort. RQUAD uses a combination of absolute and relative error criteria. The use of relative error means that RQUAD must perform an initial rough integration that adds overhead. QUAD should thus be used in place of RQUAD when the user is able to estimate the value of the integral.

SLATEC

DQAGP
Calculates an approximation result to a given definite integral $I = \int_a^b f(x) \, dx$, attempting to satisfy the following accuracy criterion:
$$|I - \text{result}| \leq \max(|\epsilon_{\text{abs}}|, |\epsilon_{\text{rel}}| |I|).$$
Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities, discontinuities), are provided by the user. Double precision version.

DQAGPE
Calculates an approximation result to a given definite integral $I = \int_a^b f(x) \, dx$, attempting to satisfy the following accuracy criterion:
$$|I - \text{result}| \leq \max(|\epsilon_{\text{abs}}|, |\epsilon_{\text{rel}}| |I|).$$
Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities, discontinuities), are provided by the user. Provides more information than DQAGP. Double precision version.

DQAGS
Calculates an approximation result to a given definite integral $I = \int_a^b f(x) \, dx$, attempting to satisfy the following accuracy criterion:
$$|I - \text{result}| \leq \max(|\epsilon_{\text{abs}}|, |\epsilon_{\text{rel}}| |I|).$$
Designed to handle integrands with endpoint singularities. Double precision version.

DQAGSE
Calculates an approximation result to a given definite integral $I = \int_a^b f(x) \, dx$, attempting to satisfy the following accuracy criterion:
$$|I - \text{result}| \leq \max(|\epsilon_{\text{abs}}|, |\epsilon_{\text{rel}}| |I|).$$
Designed to handle integrands with endpoint singularities. Provides more information than DQAGS. Double precision version.

QAGP
Calculates an approximation result to a given definite integral $I = \int_a^b f(x) \, dx$, attempting to satisfy the following accuracy criterion:
$$|I - \text{result}| \leq \max(|\epsilon_{\text{abs}}|, |\epsilon_{\text{rel}}| |I|).$$
Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities, discontinuities), are provided by the user.
QAGPE
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]
Break points of the integration interval, where local difficulties of the integrand may occur (e.g. singularities, discontinuities), provided by user. Provides more information than QAGF.

QAGS
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]
Designed to handle integrands with endpoint singularities.

QAGSE
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) \, dx \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]
Designed to handle integrands with endpoint singularities. Provides more information than QAGS.

H2a3b1. One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand

NAG

D01ANF
Calculates an approximation to the cosine or the sine transform of a function \( G \) over \((A, B)\), i.e. \( \int_A^B G(x) \cos(\omega x) \, dx \) or \( \int_A^B G(x) \sin(\omega x) \, dx \) (for a user-specified value of \( \omega \)).

D01APF
Estimates the value of a definite integral over a finite range, with an algebraic or logarithmic weight function, to a specified absolute or relative accuracy, using Gauss-Kronrod and modified Clenshaw-Curtis rules in an adaptive strategy.

SLATEC

DQAWO
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) w(x) \, dx \) where \( w(x) = \cos(\omega x) \) or \( w(x) = \sin(\omega x) \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]
Double precision version.

DQAWOWE
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) w(x) \, dx \) where \( w(x) = \cos(\omega x) \) or \( w(x) = \sin(\omega x) \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]
Provides more information than DQAWO. Double precision version.

QAWO
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) w(x) \, dx \) where \( w(x) = \cos(\omega x) \) or \( w(x) = \sin(\omega x) \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]

QAWOE
Calculates an approximation result to a given definite integral \( I = \int_a^b f(x) w(x) \, dx \) where \( w(x) = \cos(\omega x) \) or \( w(x) = \sin(\omega x) \), attempting to satisfy the following accuracy criterion:
\[
|I - \text{result}| \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} | I |)
\]

H2a3b2. One-Dimensional Integration: Finite Interval, Automatic, Weighted Integrand with Singularities or Discontinuities

**NAG**

**D01AQF** Calculates an approximation to the Hilbert transform of a function $G(x)$ over $(A,B)$, i.e., $\int_{A}^{B} \frac{G(x)}{x-C} dx$, for user-specified values of $A,B,C$.

**SLATEC**

**DQAWC** Calculates an approximation result to a Cauchy principal value $I = \int_{a}^{b} f(x)w(x)dx$ ($w(x) = \frac{1}{x-c}$, $c \neq a, c \neq b$), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$. Double precision version.

**DQAWCE** Calculates an approximation result to a Cauchy principal value $I = \int_{a}^{b} f(x)w(x)dx$ ($w(x) = \frac{1}{x-c}$, $c \neq a, c \neq b$), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$. Provides more information than DQAWC. Double precision version.

**DQAWS** Calculates an approximation result to a given definite integral $I = \int_{a}^{b} f(x)w(x)dx$, (where $w$ shows a singular behavior at the end points), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$. Double precision version.

**DQAWSE** Calculates an approximation result to a given definite integral $I = \int_{a}^{b} f(x)w(x)dx$, (where $w$ shows a singular behavior at the end points), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$. Provides more information than DQAWS. Double precision version.

**QAWC** Calculates an approximation result to a Cauchy principal value $I = \int_{a}^{b} f(x)w(x)dx$ ($w(x) = \frac{1}{x-c}$, $c \neq a, c \neq b$), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$.

**QAWCE** Calculates an approximation result to a Cauchy principal value $I = \int_{a}^{b} f(x)w(x)dx$ ($w(x) = \frac{1}{x-c}$, $c \neq a, c \neq b$), attempting to satisfy the following criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$. Provides more information than QAWC.

**QAWS** Calculates an approximation result to a given definite integral $I = \int_{a}^{b} f(x)w(x)dx$, (where $w$ shows a singular behavior at the end points), attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \text{max}(e_{abs}, e_{rel} | I |)$.
QAWSE  Calculates an approximation result to a given definite integral
$I = \int_a^b f(x)w(x)dx$, (where $w$ shows a singular behavior at the end points),
attempting to satisfy the following accuracy criterion:
$|I - \text{result}| \leq \max(e_{abs}, e_{rel} |I |)$. Provides more information than QAWS.

H2a4. One-Dimensional Integration: Infinite Interval

NAG

D01AMF  Calculates an approximation to the integral of a function $F(x)$ over an infinite or
semi-infinite interval $(A,B)$.

D01BAF  Computes an estimate of the definite integral of a function of known analytical
form, using a Gaussian quadrature formula with a specified number of abscissae.
Formulae are provided for a finite interval (Gauss-Legendre), a semi-infinite interval
(Gauss-Laguerre, Gauss-Rational), and an infinite interval (Gauss-Hermite).

SLATEC

DQAGI  Calculates an approximation result to a given integral $I = \int_{-\infty}^{+\infty} f(x)dx$ or
$I = \int_{-\infty}^{+\infty} f(x)dx$ or $I = \int_{-\infty}^{+\infty} f(x)dx$, attempting to satisfy the following accuracy
criterion: $|I - \text{result}| \leq \max(e_{abs}, e_{rel} |I |)$. Double precision version.

DQAGIE  Calculates an approximation result to a given integral $I = \int_{-\infty}^{+\infty} f(x)dx$ or
$I = \int_{-\infty}^{+\infty} f(x)dx$ or $I = \int_{-\infty}^{+\infty} f(x)dx$, attempting to satisfy the following accuracy
criterion: $|I - \text{result}| \leq \max(e_{abs}, e_{rel} |I |)$. Provides more information than DQAGI. Double precision version.

DQAWF  Calculates an approximation result to a given Fourier integral
$I = \int_{-\infty}^{+\infty} f(x)w(x)dx$ where $w(x) = \cos(\omega x)$ or $w(x) = \sin(\omega x)$, attempting to
satisfy the following accuracy criterion: $|I - \text{result}| \leq e_{abs}$. Double precision version.

DQAFWE  Calculates an approximation result to a given Fourier integral
$I = \int_{-\infty}^{+\infty} f(x)w(x)dx$ where $w(x) = \cos(\omega x)$ or $w(x) = \sin(\omega x)$, attempting to
satisfy the following accuracy criterion: $|I - \text{result}| \leq e_{abs}$. Provides more
information than DQAWF. Double precision version.

DQK15I  The original (infinite) integration range is mapped onto the interval $(0,1)$ and
(A,B) is a part of $(0,1)$. The purpose is to compute $I = \text{Integral of transformed}
\text{integrand over (A,B)}, J = \text{Integral of ABS(Transformed Integrand) over (A,B)}$

QAGI  Calculates an approximation result to a given integral $I = \int_{-\infty}^{+\infty} f(x)dx$ or
$I = \int_{-\infty}^{+\infty} f(x)dx$ or $I = \int_{-\infty}^{+\infty} f(x)dx$ attempting to satisfy the following accuracy
criterion: $|I - \text{result}| \leq \max(e_{abs}, e_{rel} |I |)$.

QAGIE  Calculates an approximation result to a given integral $I = \int_{-\infty}^{+\infty} f(x)dx$ or
$I = \int_{-\infty}^{+\infty} f(x)dx$ or $I = \int_{-\infty}^{+\infty} f(x)dx$, attempting to satisfy the following accuracy
criterion: $|I - \text{result}| \leq \max(\epsilon_{abs}, \epsilon_{rel}, |I|)$ Provides more information than QAGI.

QAWF
Calculates an approximation result to a given Fourier integral $I = \int_{\omega}^{\infty} f(x)w(x)dx$ where $w(x) = \cos(\omega x)$ or $w(x) = \sin(\omega x)$, attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \epsilon_{obs}$.

QAWFE
Calculates an approximation result to a given Fourier integral $I = \int_{\omega}^{\infty} f(x)w(x)dx$ where $w(x) = \cos(\omega x)$ or $w(x) = \sin(\omega x)$, attempting to satisfy the following accuracy criterion: $|I - \text{result}| \leq \epsilon_{obs}$. Provides more information than QAWF.

QK15I
The original (infinite) integration range is mapped onto the interval (0,1) and $(A,B)$ is a part of (0,1). The purpose is to compute $I = \text{Integral of transformed integrand over (A,B)}$, $J = \text{Integral of ABS(Transformed Integrand) over (A,B)}$, using a 15-point Kronrod rule.

H2b. Multi-Dimensional Integration

FITPACK

AREAKP
Determines the area enclosed by a closed curve in the plane. The sign associated with the area is such that the area is positive if the curve goes around the interior in a counter-clockwise fashion (and negative if clockwise). The subroutine KURVP1 should be called earlier to determine necessary parameters.

SURFI
Integrates a surface specified by a spline under tension within a rectangle. The subroutine SURF1 should be called earlier to determine certain necessary parameters.

IMSL

DBCQDU
Bicubic spline quadrature.

DBLIN
Numerical integration of a function of two variables.

DMLIN
Numerical integration of a function of several variables over a hyper-rectangle (Gaussian method).

NAG

D01DAF
Attempts to evaluate a double integral to a specified absolute accuracy by repeated applications of the method described by Patterson.

D01FBF
Computes an estimate of a multidimensional integral (from 1 to 20 dimensions), given the analytic form of the integrand and suitable Gaussian weights and abscissae.

D01FCF
Attempts to evaluate a multidimensional integral (up to 15 dimensions), with constant and finite limits, to a specified relative accuracy, using an adaptive subdivision strategy.
D01FDF Calculates an approximation to a definite integral of up to 30 dimensions using the method of Sag and Szekeres. The region of integration is the \( n \)-sphere or, by built-in transformation via the unit \( n \)-cube, any product region.

D01GBF Calculates an approximation to the integral of a function defined over a hyper rectangle, by a Monte-Carlo method, and returns an error estimate. (this routine replaced D01FAF.)

D01GCF Calculates an approximation to a definite integral of up to 20 dimensions using a stochastic form of the Korobov-Conroy number-theoretic method.

D01GYF Calculates the Korobov optimal coefficients, for use in D01GCF, when the number of points is a prime integer.

D01GZF Calculates the Korobov optimal coefficients, for use in D01GCF, when the number of points is a product of two prime integers.

D01JAF Calculates an approximation to a definite integral over an \( n \)-sphere, for \( n = 2, 3 \) or 4, to a user-specified absolute or relative accuracy, by a modification of the method of Sag and Szekeres; this routine can handle singularities on the surface or at the centre of the sphere, and returns an error estimate.

D01PAF Calculates a sequence of approximations to the integral of a function over a multi-dimensional simplex, together with an error estimate for the last approximation.

H2c. Computation of Weights and Nodes for Quadrature (Integration) Formula

**NAG**

**D01BBF** Returns the weights and abscissae appropriate to a Gaussian quadrature formula with a specified number of abscissae. The formulae provided are Gauss-Legendre, Gauss-Rational, Gauss-Laguerre and Gauss-Hermite.

**D01BCF** Returns the weights (normal or adjusted) and abscissae for a Gaussian integration rule with a specified number of abscissae. Six different types of Gauss rule are allowed.

**NCARLB**

**GAUSL** Calculates Gauss-Legendre weights and abscissae on a given finite interval or integrates a given function using Gauss-Legendre quadrature.

**GAUSS** Calculates Gaussian quadrature abscissas and weights relative to a given weight function on a given finite or infinite interval of integration.
PORT

DGAUSQ  Double precision version of GAUSQ.
DGQOIN  Double precision version of GQOIN.
GAUSQ   Computes the abscissae, $x_i$, and weights, $w_i$, for an $N$-point Gauss quadrature rule, given coefficients for a recurrence relation defining a set of polynomials over the integration interval, and moments of the polynomials integrated with a specified weight function.
GQOIN   Computes the abscissae and weights for an $N$-point Gauss-Laguerre quadrature rule on the interval $(0, +\infty)$.
GQM11   Computes the abscissae and weights for an $N$-point Gauss-Legendre quadrature rule on the interval $(-1, 1)$.

SLATEC

DQMOMO  This routine computes modified Chebyshev moments. The $K^{th}$ modified Chebyshev moment is defined as $\int_{-1}^{1} w(x) T_K(x) dx$, where $T_K(x)$ is the Chebyshev polynomial of degree $K$. Double precision version.
QMOMO   This routine computes modified Chebyshev moments. The $K^{th}$ modified Chebyshev moment is defined as $\int_{-1}^{1} w(x) T_K(x) dx$, where $T_K(x)$ is the Chebyshev polynomial of degree $K$. 

Differentiation, Integration
Ila. Optimization (Function Minimization): One-Dimensional

**IMSL**

ZXGSN  One-dimensional unimodal function minimization using the golden section search method.

ZXGSP  One-dimensional unimodal function minimization using the golden section search method – data parameters specified.

ZXLSF  One-dimensional minimization of a smooth function using safeguarded quadratic interpolation.

**NAG**

E04ABF  Searches for a minimum, in a given finite interval, of a continuous function of a single variable, using function values only. The method (based on quadratic interpolation) is intended for functions which have a continuous first derivative (although it will usually work if the derivative has occasional discontinuities).

E04BBF  Searches for a minimum, in a given finite interval, of a continuous function of a single variable, using function and first derivative values. The method (based on cubic interpolation) is intended for functions which have a continuous first derivative (although it will usually work if the derivative has occasional discontinuities).

**PORT**

DFMIN  Double precision version of FMIN.

EXTRMD  Finds extremal points of a double precision-valued discrete function defined on a mesh. It does so by dividing the domain into the largest intervals on which the function does not change sign and then on each interval it finds a point of maximum magnitude.

EXTRMI  Finds extremal points of an integer-valued function defined on a discrete mesh. It does so by dividing the domain into the largest intervals on which the function does not change sign and then on each interval it finds a point of maximum magnitude.

EXTRMR  Finds extremal points of a real-valued function defined on a discrete mesh. It does so by dividing the domain into the largest intervals on which the function does not change sign and then on each interval it finds a point of maximum magnitude.

FMIN  Local minimum of a function. Finds an approximation \( x \) to the point in an interval, specified by the user, at which \( f(x) \) attains a minimum. If \( f \) is not unimodal within the interval, FMIN finds a local minimum. Brent's algorithm, a combination of golden section search and successive parabolic interpolation, is used.
I1b. Optimization (Function Minimization): Multi-Dimensional, Unconstrained

IMSL

ZSRCH
Generate starting points in an $n$-dimensional space, for non-linear optimization problems.

ZXCGR
A conjugate gradient algorithm for finding the minimum of a function of $n$ variables.

ZXMIN
Minimum of a function of $n$ variables using a quasi-Newton method.

NAG

E04CCF
Minimizes a general function $F(X)$ of $N$ independent variables by the Simplex method. Derivatives of the function need not be supplied.

E04CGF
Is an easy-to-use quasi-Newton algorithm for finding an unconstrained minimum of a function of $N$ independent variables using function values only. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04DBF
Minimizes a general function $F(X)$ of $N$ independent variables by the conjugate gradient method due to Fletcher and Reeves. Formulae to calculate the value of the function and its first derivatives must be supplied.

E04DEF
Is an easy-to-use quasi-Newton algorithm for finding an unconstrained minimum of a function of $N$ independent variables when first derivatives are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04DFF
Is an easy-to-use modified-Newton algorithm for finding an unconstrained minimum of a function of $N$ independent variables when first derivatives are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04EBF
Is an easy-to-use modified-Newton algorithm for finding an unconstrained minimum of a function of $N$ independent variables when first and second derivatives are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04JBF
Is a comprehensive quasi-Newton algorithm for finding either an unconstrained minimum of a function of several variables or a minimum of a function of several variables subject to fixed upper and/or lower bounds on the variables. No derivatives are required. The routine is intended for functions which have continuous first and second derivatives (although it will usually work even if the
derivatives have occasional discontinuities).

**E04KBF**
Is a comprehensive quasi-Newton algorithm for finding either an unconstrained minimum of a function of several variables or a minimum of a function of several variables subject to a fixed upper and/or lower bounds on the variables. First derivatives are required. The routine is intended for functions which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

**E04KDF**
Is a comprehensive modified Newton algorithm, using first derivatives, for finding:
- an unconstrained minimum of a function of several variables
- a minimum of a function of several variables subject to simple bounds.

**E04LBF**
Is a comprehensive modified Newton algorithm, using first and second derivatives, for finding either an unconstrained minimum of a function of several variables or a minimum of a function of several variables subject to simple bounds.

---

**I2a. Optimization (Function Minimization): Multi-Dimensional, Linear and Quadratic Programming**

**IMSL**

**ZX0LP**
Solve the linear programming problem (phase one or phase two) via the revised simplex algorithm.

**ZX3LP**
Solve the linear programming problem via the revised simplex algorithm – easy to use version.

**ZX4LP**
Solve the linear programming problem via the revised simplex algorithm (alternate easy to use version).

**NAG**

**E04MBF**
Is an easy-to-use routine for solving linear programming problems, or for finding a feasible point for such problems. It is not intended for sparse problems.

**E04NAF**
Is a comprehensive routine for solving quadratic programming (QP) or linear programming (LP) problems. It is not intended for sparse problems.

**H01ABF**
Solves the linear programming problem via the simplex algorithm. The subroutine only handles inequality constraints. By repeated calls successive tableaux can be obtained.

**H01ADF**
Solves the linear programming problem via the revised simplex method.

**H01AFF**
Finds a point or, optionally, a vertex which satisfies a given set of linear constraints.

**H01BAF**
Is a comprehensive routine for solving the linear programming problem. It uses a numerically stable form of the simplex method.

**H02AAF**
Minimises a symmetric, positive-definite quadratic form in \( n \) non-negative variables, subject to \( m \) linear constraints which may be equations or upper-bound constraints.
inequalities.

**H02BAF**
Solves the integer linear programming problem, with all integer coefficients, via Gomory's method. It is enhanced by including the technique known as Wilson's cuts.

**H03ABF**
Solves the classical transportation ("Hitchcock") problem.

---

**SLATEC**

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

---

**I2h1. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Linear or Bounds Constraints**

---

**IMSL**

**ZXMWD**
Global minimum (with constraints) of a function of \( n \) variables.

---

**NAG**

**E04JAF**
Is an easy-to-use quasi-Newton algorithm for finding a minimum of a function, subject to fixed upper and lower bounds of the independent variables, using function values only.

**E04JBF**
Is a comprehensive quasi-Newton algorithm for finding either an unconstrained minimum of a function of several variables or a minimum of a function of several variables subject to fixed upper and/or lower bounds on the variables. No derivatives are required. The routine is intended for functions which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

**E04KAF**
Is an easy-to-use quasi-Newton algorithm for finding a minimum of a function subject to fixed upper and lower bounds on the independent variables when first derivatives of \( F \) are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

**E04KBF**
Is a comprehensive quasi-Newton algorithm for finding either an unconstrained minimum of a function of several variables or a minimum of a function of several variables subject to a fixed upper and/or lower bounds on the variables. First derivatives are required. The routine is intended for functions which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

**E04KCF**
Is an easy-to-use modified-Newton algorithm for finding a minimum of a function, subject to fixed upper and lower bounds on the independent variables, when first derivatives of \( F \) are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).
E04KDF is a comprehensive modified Newton algorithm, using first derivatives, for finding:
- an unconstrained minimum of a function of several variables - a minimum of a function of several variables subject to simple bounds.

E04LAF is an easy-to-use modified-Newton algorithm for finding a minimum of a function, subject to fixed upper and lower bounds on the independent variables, when first and second derivatives of $F$ are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04LBF is a comprehensive modified Newton algorithm, using first and second derivatives, for finding either an unconstrained minimum of a function of several variables or a minimum of a function of several variables subject to simple bounds.

I2h3. Optimization (Function Minimization): Multi-Dimensional, Nonlinear, Nonlinear Constraints

LOCLIB

VMCON is a package for calculating the least value of a function of several variables subject to linear and/or nonlinear equality and inequality constraints. The user must provide a function which calculates the objective and constraint functions, and the gradients (first derivative vectors) of the objective and constraint functions.

NAG

E04UAF attempts to find a minimum of a function of several variables subject to fixed bounds on the variables and to general equality and/or inequality constraints. A sequential augmented Lagrangian method is used, the minimization subproblems involved being solved by a quasi-Newton method. No derivatives are required. The routine is intended for functions and constraints which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04VAF attempts to find a minimum of a function of several variables subject to fixed bounds on the variables and to general equality and/or inequality constraints. A sequential augmented Lagrangian method is used, the minimization subproblems involved being solved by a quasi-Newton method. First derivatives are required. The routine is intended for functions and constraints which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

E04VBF attempts to find a minimum of a function of several variables subject to fixed bounds on the variables and to general equality and/or inequality constraints. A sequential augmented Lagrangian method is used, the minimization subproblems involved being solved by a modified Newton method. First derivatives are required. The routine is intended for functions and constraints which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).
<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E04VCF</td>
<td>Is a comprehensive routine designed to minimize an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth non-linear constraints. It is not intended for sparse problems.</td>
</tr>
<tr>
<td>E04VDF</td>
<td>Is an easy-to-use routine designed to minimize an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth non-linear constraints. It is not intended for sparse problems.</td>
</tr>
<tr>
<td>E04WAF</td>
<td>Attempts to find a minimum of a function of several variables subject to fixed bounds on the variables and to general equality and/or inequality constraints. A sequential augmented Lagrangian method is used, the minimization subproblems involved being solved by a modified Newton method. First and second derivatives are required. The routine is intended for functions and constraints which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).</td>
</tr>
</tbody>
</table>
J1. Sequential Input/Output, Including Tape I/O

LOCLIB

CORFOR Provides facility for copying FORTRAN-written binary tapes with bad parity records copied or deleted.

UBLOK Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries.

UZBLOK Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries.

NCARLB

IOPROC A collection of subroutines RDTAPE, WRTAPE, and IOWAIT, for asynchronous I/O.

RUGRID Unpacks and reads National Meteorological Center grid data.

SYSLIB

BKSP Contains the backspacing routines $BKSP and $BKSPF. $BKSP backspaces one record. The function is nonoperational if the dataset is at the beginning of a dataset. If the dataset is at the first record of a file, this routine positions the dataset ahead of the EOF. $BKSPF backspaces one file.

EOADF Dataset termination routines for writing EOD, and also EOF and EOR if necessary; also, clears UEOF flag in DSP.

EODF Writes an end-of-data, also an end-of-file and end-of-record if necessary.

EOFR Contains $EOFR, callable from CAL only, which determines if UEOF flag in DSP (dataset parameter area) is set and clears the UEOF flag in the DSP.

EOWF Writes an EOF, also an EOR if necessary, and clears the unchecked EOF flag in the DSP.

OPEN Connects existing dataset to unit, creates preconnected dataset, creates dataset and connects it to unit, or changes certain specifiers of connection between dataset and unit.

SYNCH Allows a CAL user to synchronize the program and an opened tape dataset. If the dataset is synchronized for input, the tape must be positioned at an EOR control word. However, an EOR is added to the end of the data before synchronization if the dataset is an output dataset and the data in the circular buffer does not end with an EOR control word. A tape dataset is not synchronized after any I/O is issued. For an output tape, control is not returned to the user until all data in the circular buffer is written to the tape.

TAPESTAT Returns tape status in the DSP to caller.
UEOFC  Clears the uncleared end-of-file flag in DSP.
UEOFK  Aborts job if uncleared end-of-file flag is set in DSP.
UEOFS  Sets the uncleared end-of-file flag in DSP.
WCH    Writes character data, partial record mode or full record mode.
WRTUTIL Provides utilities needed to support $WWD, $WCH, and $REWD.
WWD    Writes data, partial record mode, no unused bits; or writes data, partial record mode with unused bits; or writes data, full record mode, no unused bits; or writes data, full record mode with unused bits; or writes end-of-file RCW; or writes end-of-data RCW.

J2.  Direct (Random) Access Input/Output

NCARLB

RIOF    A FORTRAN package from ECMWF which provides random synchronous I/O at the record level for the Cray.

SYSLIB

GPOS    Returns current dataset position.
GTPOS   Allows a CAL user to get position information about an opened tape dataset.
OPEN    Connects existing dataset to unit, creates preconnected dataset, creates dataset and connects it to unit, or changes certain specifiers of connection between dataset and unit.
PBN     Contains $PBN, callable from CAL only, which positions a dataset to a block number and updates the DSP.
SPOS    Positions dataset to a word address.
STPOS   Allows a CAL user to position a tape dataset at a particular tape block of the dataset. Data blocks on the tape are numbered so that block number 1 is the first data block on a tape.
SYMDBC  Contains subprograms OPEN, CLOSE, and RDIN for Cray-1 dataset I/O. OPEN opens a random, unblocked dataset. CLOSE terminates processing of a random, unblocked dataset. RDIN reads one buffer of data from a random, unblocked dataset.
UBIO    An I/O package for "word addressable dataset".
WLB     Writes data directly into user area.
NCAR Software Catalog

Category J3

J3. Asynchronous (Buffered) Input/Output

NCARLB

IOPROC  A collection of subroutines RDTAPE, WRTAPE, and IOWAIT, for asynchronous I/O.

UNLCRD  Emulates the old 7600 routines UNLCRD and UNLCWT for transferring data between LCM and disk units.

SYSLIB

CBIO  Provides buffered I/O interface for CAL requests (callable from CAL only).

J4. Free Format and Data Structure Input/Output

IMSL

USTREE  Print a binary tree (which may represent the output of a clustering algorithm in chapter O).

USWBM  Print a matrix stored in band storage mode.

USWBS  Print a matrix stored in band symmetric storage mode.

USWCH  Print a complex Hermitian matrix stored in Hermitian storage mode.

USWCM  Print a complex matrix stored in full storage mode.

USWCV  Print a complex vector.

USWFM  Print a matrix stored in full storage mode.

USWFV  Print a vector.

USWSM  Print a matrix stored in symmetric storage mode.

NCARLB

READLX  Provides free format data-directed input and program control facilities.

RUGRID  Unpacks and reads National Meteorological Center grid data.
J5a1. Graphics: Plotting Dashed Lines

**LOCLIB**

**DASHLINE** Like DASHCHAR, but smaller and faster because it has no labeling capacity.

**DASHSMTH** Like DASHCHAR, but bigger and slower because lines are smoothed.

**DASHSUPR** Like DASHCHAR, but bigger and slower because lines are smoothed and crowded lines are removed.

**NCARLB**

**DASHCHAR** Software dashed line package with labeling capability.

J5a2. Graphics: Plotting Graphs and Histograms

**IMSL**

**USPLO** Printer plot of up to ten functions.

**LOCLIB**

**DASHSUPR** Like DASHCHAR, but bigger and slower because lines are smoothed and crowded lines are removed.

**NCARLB**

**AUTOGRAPH** Draws and annotates curves or families of curves. FORTRAN 77 version.

**WINDOW** Provides a clipping capability for lines extending outside a user-defined window, thus allowing part of a picture to be plotted without distortion or overwriting near the edge of the picture.
J5a3. Graphics: Representing Two-Dimensional Fields

**LOCLIB**

**CONBND**
Given a two-dimensional array of data and a pair of contour values, CONBND returns the x and y coordinates of a set of points defining the boundary of the area between the specified contour lines; these coordinates may be fed into the LOCLIB routine FILL to shade the area between the given contour levels.

**CONBNDMI**
IFTRAN source for the FORTRAN version of file CONBND on LOCLIB.

**CONRECQCK**
Like CONREC, but faster and smaller because contours are unlabeled.

**CONRECSMTH**
Like CONREC, but bigger and slower because contours are smoothed as well as labeled.

**CONRECSUPR**
Like CONREC, but bigger and slower because contours are smoothed, labeled, and crowded lines are removed.

**FILL**
A set of routines used to fill a specified area of a plot with parallel lines. The lines may be solid or dotted, and, if the latter, the dots may be arranged according to a specified pattern, with each dot being just a dot or, if desired, a specified character.

**FILLMI**
IFTRAN source for the FORTRAN version of the file FILL, on LOCLIB, which is used to fill a specified area of a plot with parallel lines.

**WNDBRB**
Draws "meteorological wind barbs" for grid of U, V data.

**NCARLIB**

**CONRAN**
Contours irregularly spaced data, labeling the contour lines. The standard version of this package requires DASHCHAR. The smoothed version consists of this file with DASHSMTH.

**CONRAQ**
Like CONRAN, but smaller and faster because it has no labeling capacity.

**CONRAS**
Like CONRAN, but bigger and slower because lines are smoothed and crowded lines are removed.

**CONREC**
Contours two-dimensional arrays, labeling the contour lines.

**HAFTON**
Halftone (gray scale) pictures from a two-dimensional array.

**STRMLN**
Plots a representation of a vector field flow of any field for which planar vector components are given on a regular rectangular lattice, displaying both field direction (via lines of flow containing arrowheads and feathers) and field magnitude (based on distance between those flow lines).

**VELVCT**
Two-dimensional velocity field displayed by drawing arrows from the data locations.
J5a4. Graphics: Representing Objects in Three-Space

**LOCLIB**

**ISOSRFHR** Iso-valued surfaces (with hidden lines removed) from a high resolution three-dimensional array.

**NCARLB**

**ISOSRF** Iso-valued surfaces (with hidden lines removed) from a three-dimensional array.

**SRFACE** Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array.

**THREED** Provides three-space line drawing capabilities, with entry points equivalent to the line drawing entry points of the NCAR system plot package.

J5a5. Graphics: Writing Characters

**LOCLIB**

**AGUPWRTX** Allows AUTOGRAPH to be used with PWRTX.

**AGUSEPWRX** Allows the old version of AUTOGRAPH to be used with PWRX.

**DASHSUPR** Like DASHCHAR, but bigger and slower because lines are smoothed and crowded lines are removed.

**PWRTXINIT** Main program that initializes the PWRTX character digitization files.

**PWRTXMX** The master version of PWRTX, from which the LOCLIB file PWRTX is created. Documentation for creation is maintained in the file.

**PWRX** Obsolete version of PWRTX, which should be used instead.

**PWRZMI** The master version of PWRZ, from which the ULIB files PWRZS, PWRZI, and PWRZT are created. Documentation for creation is maintained in the file.

**NCARLB**

**PWRTX** Plots high quality software characters.

**PWRY** Simplest software characters.

**PWRZI** Draws characters in three-space, for use with ISOSRF.

**PWRZS** Draws characters in three-space, for use with SRFACE.
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWRZT</td>
<td>Draws characters in three-space, for use with THREED.</td>
</tr>
<tr>
<td>SCROLL</td>
<td>Title-producing package used in making movies.</td>
</tr>
<tr>
<td>TPGPHC</td>
<td>Self-contained text formatting subroutine package offering 16 basic fonts, 1601 different characters, auto-justification, mixing text and plots on the same frame, ability to add new characters, and several other features.</td>
</tr>
</tbody>
</table>

**J5a6. Graphics: Mapping**

**NCARLB**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZMAP</td>
<td>Plots continental and/or United States state outlines according to one of nine projections.</td>
</tr>
</tbody>
</table>

**J5a7. Graphics: Movie Utilities**

**NCARLB**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMPLT</td>
<td>A routine intended for movie makers running on the Cray. It simplifies the process of segmenting metacode into separate volumes to be sent to the mass storage device.</td>
</tr>
<tr>
<td>FLMHDR</td>
<td>Generates a number of frames of film as a leader to a movie to allow precise alignment of the plot device and proper focusing on the projector.</td>
</tr>
<tr>
<td>SCROLL</td>
<td>Title-producing package used in making movies.</td>
</tr>
</tbody>
</table>
J5b. Graphics: Other Utilities

**IMSL**

**USBOX**  Print a boxplot \((k\) samples).

**USPC**  Print a sample probability density function, a theoretical pdf and confidence band information. Plot these on option.

**USPDF**  Plot of two sample probability distribution functions against their spectra.

**USSLF**  Print a stem and leaf display.

**LOCILIB**

**CALCMPCF**  Source for the CALCOMP metacode translator on the Cray.

**MFEDITMI**  Source for master IFTRAN which can be used to create MFEDIT on the Cray. MFEDIT is a batch NCAR metatile editor.

**MFMERGCI**  Source for MFMERGE module on the Crays. MFMERGE merges a user-specified number of NCAR metafiles into one metatile.

**MFSPLTCI**  Source for MFSPLIT module on the Crays. MFSPLIT divides an NCAR metafile into a user-specified number of metafiles of roughly equal size.

**PRSIMMI**  The master version of PRSIM, from which the Cray, and 11/70 versions may be produced. How to do this is documented in the file itself.

**NCARLB**

**MCTR.PRINT**  Translates NCAR graphics metacode to lines of characters for a printer. This makes it possible to produce low resolution printer plots for fast turnaround and debugging without changing the calls to NCAR graphics utilities in a program.

**NAMEFR**  Produces a name frame for identifying plot output.

**PLOT.8.8**  A version of the NCAR system plot package for the Cray.

**PRSIMF**  FORTRAN-callable text formatter for printer simulation on the Dicomeds. Can be used to intersperse text and plots.

**UTILITIES**

**CALCMP**  Locally developed Cray command for CALCOMP metacode translation.

**HEXER**  Locally developed Cray command which reads standard graphics metacode (as described in the NCAR Graphics Manual) and writes a formatted hexadecimal dump of the metacode, with a special carriage control character duplicated at the beginning of each line. Can be used to send graphics files to an RJE site.

**MC2PR**  Locally developed Cray command for translating NCAR graphics metacode to lines of characters for a printer. For execution on Cray machines, enabling users to produce low resolution printer plots for fast turnaround and debugging without changing the calls to NCAR graphics utilities in a program.
MFEDIT Locally developed Cray command for editing NCAR graphics metacode.

MFMERGE Locally developed Cray command for merging NCAR graphics metafiles.

MFSPLIT Locally developed Cray command for splitting an NCAR metafile into a user-specified number of metafiles of roughly equal size.

PRSIM Locally developed Cray command for printer simulation on the Dicomeds. Can be used to intersperse text and plots.

J6. Miscellaneous I/O, Including Communications, Special Devices

LOCLIB

DGMOD Enables users to modify output from the Bendix Datagrid Prepass program PREPA.

HEXER A main program that reads standard graphics metacode (as described in the NCAR Graphics Manual) and writes a formatted hexadecimal dump of the metacode, with a special carriage control character duplicated at the beginning of each line. Can be used to send graphics files to an RJE site.

PLTGRD Any output tape from the Bendix Datagrid Prepass program of PREPB may be input to PLTGRD to produce a film display of the data points on the tape, and an instruction listing for use in DGMOD.

PREPA Processes tapes from the Bendix Datagrid System digitizer into a form that can be used in FORTRAN programs that further process the data.

UNHEXER A main program that reads a file containing lines of text (produced by HEXER, for example), extracts any embedded hex-encoded metacode file, and outputs the equivalent standard binary metacode file to unit 8.

NCARLB

BRANIO Simulates read and write capabilities of the old 7600 BRANRD package.

SYSLIB

CONNECT The connect statement causes a link to be created between a Cray job and a dataset residing on a system linked to the Cray through the ISP. Only the information about the dataset and the data in the user buffer exist on the Cray side and the system linked by the ISP maintains the dataset, responding to requests from the Cray to manipulate the dataset.

ISP Creates the initial connection between a Cray user job and an ISP subsystem. ISP statement processing will generate a taskinit message which will cause an initialization of a task on the ISP subsystem. This task will correspond to the Cray user job and provide the ISP functions requested by the job.
RLB Reads data directly from user area on the Cray.

**UTILITIES**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACQUIRE</td>
<td>Cray system control statement for making a dataset permanent and accessible to a job.</td>
</tr>
<tr>
<td>DISPOSE</td>
<td>Cray system control statement for directing a dataset to the CRAY input or output queue for staging to a specified front-end computer system.</td>
</tr>
<tr>
<td>DTLIST</td>
<td>Locally developed Cray command which reports amount of writable space remaining on dedicated TBM reels.</td>
</tr>
<tr>
<td>ECH</td>
<td>VOS Software Tools command for echoing command line arguments.</td>
</tr>
<tr>
<td>MAQR</td>
<td>Locally developed Cray command which performs simultaneous multiple ACQUIREs.</td>
</tr>
<tr>
<td>MULTI</td>
<td>Locally developed Cray command for accessing libraries and system texts necessary for use and generation of multitasking code.</td>
</tr>
<tr>
<td>NETAQR</td>
<td>Locally developed Cray command which performs non-TBM acquire functions. Similar to Cray command ACQUIRE, but somewhat faster and offering better diagnostics.</td>
</tr>
<tr>
<td>NETDISP</td>
<td>Locally developed Cray command which performs non-TBM dispose functions. Similar to Cray command DISPOSE, but somewhat faster and offering better diagnostics.</td>
</tr>
<tr>
<td>PL</td>
<td>VOS Software Tools command for printing specified lines/pages in a file.</td>
</tr>
<tr>
<td>PR</td>
<td>VOS Software Tools command for paginating files to standard output.</td>
</tr>
<tr>
<td>STACK</td>
<td>Locally developed Cray command for accessing libraries and system texts necessary for use and generation of stack-based (re-entrant) code.</td>
</tr>
<tr>
<td>WRITEDS</td>
<td>Cray system control statement for initializing a blocked dataset or writing a sequential dataset.</td>
</tr>
</tbody>
</table>
**K1. File Management: Access, Deletion, and Staging**

**SYSLIB**

**ACCESS**
Associates a saved dataset with a job.

**ACQUIRE**
Acquires a front-end resident dataset, stages it to the Cray-1, and makes it available to the job.

**CONNECT**
The connect statement causes a link to be created between a Cray job and a dataset residing on a system linked to the Cray through the ISP. Only the information about the dataset and the data in the user buffer exist on the Cray side and the system linked by the ISP maintains the dataset, responding to requests from the Cray to manipulate the dataset.

**DELETE**
Removes a saved dataset from the Cray dataset catalog. The dataset remains available for the life of the job.

**DISPOSE**
Directs a Cray dataset to the specified output queue.

**FETCH**
Obtains a front-end resident dataset and makes it local to a job on a Cray computer.

**PDMPAM**
Processes public ACCESS and PERMIT mode parameters for $SYSLIB permanent dataset related routines.

**RELEASE**
Closes a dataset, releases I/O buffer space, and renders it unavailable to the job.

**SAVE**
Makes a dataset permanent. Enters the dataset's identification and location into the DSC.

**SDACCESS**
Allows a FORTRAN program to access datasets in the Cray-1 System Directory.

**UTILITIES**

**ACCESS**
Cray system control statement for making an existing permanent dataset local to a job.

**ACQUIRE**
Cray system control statement for making a dataset permanent and accessible to a job.

**DISPOSE**
Cray system control statement for directing a dataset to the CRAY input or output queue for staging to a specified front-end computer system.

**MAQR**
Locally developed Cray command which performs simultaneous multiple ACQUIREs.

**NETAQR**
Locally developed Cray command which performs non-TBM acquire functions. Similar to Cray command ACQUIRE, but somewhat faster and offering better diagnostics.

**NETDISP**
Locally developed Cray command which performs non-TBM dispose functions. Similar to Cray command DISPOSE, but somewhat faster and offering better diagnostics.
**PDEL**
Locally developed Cray command for deleting one or more files and/or directories from the user's PSTORE. This PSTORE command, and several others, are also available on NCAR’s IBM 4341 machines.

**RELEASE**
Cray system control statement for relinquishing access to the named datasets.

**REMOVE**
Locally developed Cray command to remove permanent datasets from the CRAY disks.

**RM**
VOS Software Tools command for removing files.

**SEGRLS**
Cray system control statement for releasing segment datasets.

---

**K2. File Copying, Renaming**

**UTILITIES**

**COPYD**
Cray system control statement for copying a complete dataset to another, starting at their current positions.

**COPYF**
Cray system control statement for copying a specified number of files from a dataset to another, starting at their current positions.

**COPYR**
Cray system control statement for copying a specified number of records from a dataset to another, starting at their current positions.

**COPYU**
Cray system control statement for copying an unblocked dataset to another.

**MV**
VOS Software Tools command moving (renaming) a file.

**PCOPY**
Locally developed Cray command for copying one or more files from a user's, or another user's, PSTORE to an equal number of native files. This utility also performs the inverse operation of copying native files to PSTORE. This PSTORE command, and several others, are also available on NCAR’s IBM 4341 machines.

---

**K3. File Editing**

**UTILITIES**

**CH**
VOS Software Tools command for making changes in text files.

**ED**
VOS Software Tools command for editing text files.

**EDITOR**
Locally developed Cray command for text editing.

**INCLUD**
VOS Software Tools command for file inclusion.
NCAR Software Catalog

VOS SOFTWARE TOOLS stream editor utility

File Combination, Splitting, and Conversion Tools

UTILITIES

CAT  VOS Software Tools command for concatenating and printing text files.

DETAB  VOS Software Tools command for converting tabs to spaces.

LAM  VOS Software Tools command for laminating files. That is, the first output line is the result of concatenating the first lines of each input file, and so on. Null lines are used in place of missing lines when input files are of different lengths.

MCOL  VOS Software Tools command for multicolumn formatting.

REV  VOS Software Tools command for copying the specified files to standard output, reversing the order of the characters in every line.

ROFF  VOS SOFTWARE TOOLS text-formatting utility.

SPLIT  VOS Software Tools command for splitting a file.

TAIL  VOS Software Tools command for printing the last lines of a file.

TEE  VOS Software Tools command for copying input to output and named files.

UNIQ  VOS Software Tools command for stripping adjacent repeated lines from input file(s).

File Directory, Status, and Size Information

LOCLIB

TBMVSN  Cray utility program that lists the current MSD volume names associated with a given user number or project number, and which volumes are subject to the next MSD volume purge.

SYSLIB

IFDNT  Determines if a Cray-1 dataset has been accessed.

NUMBLKS  Returns current size, in 512-word blocks, of a Cray-1 dataset.

PDDED  Returns the edition number of the last dataset that was accessed, acquired, or saved.
### UTILITIES

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT</td>
<td>Cray system control statement for reporting on the status of each specified permanent dataset.</td>
</tr>
<tr>
<td>AUDPL</td>
<td>Cray system control statement for providing information on program libraries written by UPDATE.</td>
</tr>
<tr>
<td>DTLIST</td>
<td>Locally developed Cray command which reports amount of writable space remaining on dedicated TBM reels.</td>
</tr>
<tr>
<td>PDIR</td>
<td>Locally developed Cray command for displaying the names and attributes (such as creation date) of all files in a user's, or another user's, PSTORE. This PSTORE command, and several others, are also available on NCAR's IBM 4341 machines.</td>
</tr>
<tr>
<td>SHOW</td>
<td>VOS Software Tools command for showing all characters within a file.</td>
</tr>
<tr>
<td>TBMVSN</td>
<td>Locally developed Cray command program that lists the current MSD volume names associated with a given user number or project number, and which volumes are subject to the next MSD volume purge.</td>
</tr>
<tr>
<td>WC</td>
<td>VOS Software Tools command for counting lines, words, and characters in files.</td>
</tr>
<tr>
<td>XREF</td>
<td>VOS Software Tools command for making a cross reference of symbols in files.</td>
</tr>
</tbody>
</table>

### File Comparison

<table>
<thead>
<tr>
<th>Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM</td>
<td>VOS Software Tools command for printing lines common to two files.</td>
</tr>
<tr>
<td>COMPARE</td>
<td>Cray system control statement for listing differences between two blocked datasets.</td>
</tr>
<tr>
<td>DIFF</td>
<td>VOS Software Tools command for isolating differences between files.</td>
</tr>
</tbody>
</table>
K7. File Archiving and Version Control

**LOCLIB**

TBMCONV: Cray utility program that reads TBM volumes written by the CDC 7600, including volumes disposed from the Cray to the TBM via the 7600. TBMCONV unpacks the original records, converts them to Cray records, and writes the resulting records to a user-specified Cray dataset.

TBMVSN: Cray utility program that lists the current MSD volume names associated with a given user number or project number, and which volumes are subject to the next MSD volume purge.

UPKEEP: Cray utility program for TBM volume maintenance; useful for purge protection and deletion; driven by user-specified input dataset of volume names to be protected or deleted.

**UTILITIES**

ADMIN: VOS Software Tools command for administering files in tools history file format.

AR: VOS Software Tools command for maintaining archive files.

GET: VOS Software Tools command for retrieving earlier versions of text maintained in tools history file format.

PCOPY: Locally developed Cray command for copying one or more files from a user's, or another user's, PSTORE to an equal number of native files. This utility also performs the inverse operation of copying native files to PSTORE. This PSTORE command, and several others, are also available on NCAR's IBM 4341 machines.

PDEL: Locally developed Cray command for deleting one or more files and/or directories from the user's PSTORE. This PSTORE command, and several others, are also available on NCAR's IBM 4341 machines.

PDIR: Locally developed Cray command for displaying the names and attributes (such as creation date) of all files in a user's, or another user's, PSTORE. This PSTORE command, and several others, are also available on NCAR's IBM 4341 machines.

PDSLOAD: Cray system control statement for loading permanent datasets from a dataset created by PDSDUMP.

UPDATE: Cray system control statement for loading the UPDATE program into the user field and beginning execution.

UPKEEP: Locally developed Cray command for purge-maintenance of TBM volumes.
K8. File Positioning

**SYSLIB**

**REW**D
Rewinds Cray dataset.

**UTILITIES**

**SKIPD**
Cray system control statement for positioning a dataset at end of data.

**SKIPF**
Cray system control statement for skipping a specified number of files from the current position of the named dataset.

**SKIPR**
Cray system control statement for skipping a specified number of records from the current position of the named dataset.

**SKIPU**
Cray system control statement for skipping a number of sectors or all data from current position to named dataset.

K9. File Characteristics and Permissions (Access Control)

**SYSLIB**

**ADJUST**
Expands or contracts a permanent dataset.

**ALF**
Adds name to logical file table.

**ASSIGN**
Opens a dataset for reading and writing and assigns characteristics to it.

**MODIFY**
Changes permanent dataset characteristics.

**PDD**
A table of Cray-1 permanent dataset definitions that is created and managed by the dataset management subprograms, but can be referenced from a FORTRAN program.

**PDMADN**
Processes the ADN and ADN propagation attributes for permanent dataset related routines in $SYSLIB$.

**PDMPAM**
Processes public ACCESS and PERMIT mode parameters for $SYSLIB$ permanent dataset related routines.

**PDMTA**
Processes the track accesses parameter for permanent dataset related $SYSLIB$ routines.

**PERMIT**
Allows the owner of a permanent dataset to control the manner in which other users can use the dataset.
UTILITIES

ASSIGN  Cray system control statement for assigning dataset characteristics.

MODIFY  Cray system control statement for modifying permanent dataset information established by the SAVE function or previously executed MODIFY function.

PERMIT  Cray system control statement for granting/denying specified users access to a permanent dataset.

SAVE  Cray system control statement for making a local dataset permanent.

L01a.  Descriptive Statistics: Location, Dispersion, Shape

EDA

LVALS  For a batch of values, finds selected quantiles known as letter values.

IMSL


BECOVVM  Means and variance-covariance matrix.

BECVL  Variances and covariances of linear functions (out-of-core version).

BECVLI  Variances and covariances of linear functions.

BEGRPS  Moments estimation for grouped data with and without Sheppard's corrections.

BEIGRP  Estimation of basic statistical parameters using grouped data.

BEIUGR  Estimation of basic statistical parameters using ungrouped data.

BEMMI  Estimates of means, standard deviations, correlation coefficients, and third and fourth moments from a data matrix containing missing observations. (in-core version)

BEMMO  Estimates of means, standard deviations, correlation coefficients, and third and fourth moments from a data matrix containing missing observations. (out of core version)

BESTAT  Computations of basic univariate statistics from data possibly containing missing values, with weighting on option.

NMTIE  Tie statistics, given a sample of observations.
NAG

GO1AAF Calculates the mean, standard deviation, coefficients of skewness and kurtosis, and the maximum and minimum values for a set of ungrouped data. Weighting may be used.

GO1ABF Computes the means, standard deviations, corrected sums of squares and products, maximum and minimum values, and the product-moment correlation coefficient for two variables. Unequal weighting may be given.

GO1ADF Calculates the mean, standard deviation and coefficients of skewness and kurtosis for data grouped on a frequency distribution.

SAS (IBM 4341)


DATACHK For each numeric variable in an input data set, DATACHK prints the number of nonmissing observations, the five smallest and the five largest values, and the number of observations having the six prespecified missing values. Documented in SUGI Supplemental Library User's Guide - 1983 Edition.


MEANS Produces simple univariate descriptive statistics for numeric variables, including number of observations, number of missing values, mean, standard deviation, smallest value, largest value, range, sum, variance, uncorrected sum of squares, corrected sum of squares, standard error of the mean, coefficient of variation, measure of skewness, measure of kurtosis, Student's \( t \) value for testing the hypothesis that the population mean is zero, the probability of a greater absolute value of Student's \( t \), and the sum of the WEIGHT variable values. Observations may be weighted, printing may be suppressed, and selected statistics may be output to a SAS data set. Missing values are excluded before calculating statistics. Documented in SAS User's Guide: Basics - Version 5.


SUMMARY Computes descriptive statistics on numeric variables in a SAS data set and outputs the results to a new SAS data set. By using the CLASS statement the user can name the variables to be used to form subgroups, so statistics can be calculated for each subgroup as well as for the whole sample. Statistics computed can include the number of observations in the subgroup with nonmissing values.
number of observations in the subgroup having missing values for the variable, as well as all the statistics produced by PROC MEANS. Missing values can be ignored or treated as valid observations. Observations may be weighted. Documented in *SAS User's Guide: Basics - Version 5*.

**TABLES**

Computes and tabulates simple univariate statistics for all numeric variables for a control group and up to ninety-nine treated groups. Documented in “Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131”.

**TABULATE**

Constructs tables of descriptive statistics from compositions of classification variables, analysis variables, and statistics keywords. Tables can have up to three dimensions: column, row, and page. Includes the same statistics as PROC MEANS and, in addition, variance, the percentage of frequency, and the percentage of SUM. Missing values can be ignored or treated as valid observations. Observations may be weighted. Documented in *SAS User's Guide: Basics - Version 5*.

**UNIVARIATE**

Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and-leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of the weights, largest value, smallest value, range, mode, and signed rank statistic. Observations may be weighted and missing values can be excluded. Documented in *SAS User's Guide: Basics - Version 5*.

**USS**


**VAR**


**STATLIB**

Provides extensive statistical analysis of a single random variable. Included in the automatic printout are 53 statistics which summarize the sample through measures of location, measures of dispersion, and diagnostic features such as tests for outliers, non-normality, trends, and non-randomness. Common statistics such as Student's *t* and confidence intervals for the mean and standard deviation are also included.

**STATSS**

Is a statistical analysis routine for a single random variable. It is identical to STATLIB routine STATS except that it performs weighted analysis and allows optional printout.

**STATSW**

Is a statistical analysis routine for a single random variable. It is identical to STATLIB routine STATS except that different weights can be given to each observation.
L01b. Descriptive Statistics: Distributions, Densities, Frequencies (see also L03a and L03e)

**EDA**

**RGCOMP** Perform computations for a suspended rootogram.

**RGPRNT** Print, bin by bin, the observed count, the raw residual, the double-root residual, and an abbreviated display of the double-root residual.

**RGFREQ** Given a batch of values in increasing order, and a set of bin boundaries, determine the bin counts.

**IMSL**

**BDCOU1** Tally of observations into a one-way frequency table.

**BDCOU2** Tally of observations into a two-way frequency table.

**BDLTV** Produce letter value summary.

**BDTAB** Computations of frequencies of multivariate data.

**BDTWT** Computations of a two-way frequency table.

**NDKER** Nonparametric probability density function (one dimensional) estimation by the kernel method.

**NDMPLE** Nonparametric probability density function (one dimensional) estimation.

**NMRANK** Numerical ranking.

**NAG**

**G01AEF** Constructs a frequency distribution of a variable, according to either user-supplied, or routine-calculated class boundary values.

**SAS (IBM 4341)**

**FREQ** Produces one-way to n-way frequency and crosstabulation tables. For two-way tables, PROC FREQ computes tests and measures of association. For n-way tables, PROC FREQ does stratified analysis, computing statistics within as well as across strata. Frequencies can also be output to a SAS data set. Handles missing values and weighted observations. Documented in *SAS User's Guide: Basics - Version 5*.

**PCTL** Computes percentiles for one or more numeric variables and outputs them to a SAS data set. Documented in *SUGI Supplemental Library User's Guide - 1983 Edition*.

**UNIVARIATE** Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and-leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of the weights, largest value, smallest value, range, mode, and signed rank statistic.
Observations may be weighted and missing values can be excluded. Documented in *SAS User's Guide: Basics - Version 5.*

L02. Data Manipulation

**IMSL**

BDTRGI Trans-generation of the columns of a matrix (in-core version).

BDTRGO Trans-generation of the columns of a matrix (out-of-core version).

GGPER Generate a random permutation of the integers 1 to \( k \).

**NAG**

G02CEF Takes selected elements from two vectors (typically vectors of means and standard deviations) to form two smaller vectors, and selected rows and columns from two matrices (typically either matrices of sums of squares and cross-products of deviations from means and Pearson product-moment correlation coefficients, or matrices of sums of squares and cross-products about zero and correlation-like coefficients) to form two smaller matrices, allowing re-ordering of elements in the process.

G02CFF Re-orders the elements in two vectors (typically vectors of means and standard deviations), and the rows and columns in two matrices (typically either matrices of sums of squares and cross-products of deviations from means and Pearson product-moment correlation coefficients, or matrices of sums of squares and cross-products about zero and correlation-like coefficients).

G05EHF Generates a pseudo-random permutation of a vector.

G05EJF Generates a pseudo-random subset of a vector.

**SAS (IBM 4341)**

COMPUTAB Produces reports in a row and column arrangement. With COMPUTAB (computing and tabular reporting) you can define the format of a table, label the table, and operate on its row and column values. Documented in *SAS/ETS User's Guide - Version 5.*


MATRIX An interpretive programming language in which operations are performed on entire matrices of values. Besides defining the common matrix operations such as multiple, transpose, and direct product, many functions and commands are also defined, such as, eigenvalues, generalized inverse, Cholesky decomposition and singular value decomposition. Documented in "The MATRIX Procedure: Language and Application, Technical Report: P-135".
SORT  Sorts observations in a SAS data set by one or more variables, storing the resulting sorted observations in a new SAS data set or replacing the original. Options documented in *SAS User's Guide: Basics - Version 5*.

STANDARD  Standardizes some or all of the variables in a SAS data set to a given mean and standard deviation and produces a new SAS data set to contain the standardized values. Documented in *SAS User's Guide: Statistics - Version 5*.


L03a.

**Statistical Graphics: Histograms**

**IMSL**

USHHST  Print a horizontal histogram.

USHST   Print a vertical histogram.

USHST2  Print a vertical histogram, plotting two frequencies with one bar of the histogram.

**NAG**

G01AJF  Prints a histogram of one variable.

**SAS (IBM 4341)**

CHART  Produces vertical and horizontal bar charts, block charts, pie charts, and star charts. Options include making each bar or section represent the percentage of observations, the frequency of observations, the cumulative percentage of observations, the sum of observations by another variable or the mean of observations by another variable. Additional options documented in *SAS User's Guide: Basics - Version 5*.

**STATLIB**

HISTO  Produces line-printer histograms, using a preset procedure for choosing the number of cells and “nice” cell midpoints. Recommended when the data structure and the number of cells is unknown.

HISTOC  Produces line-printer histograms, allowing the user to specify the number of cells and the upper and lower boundaries of the cells.
L03b. Statistical Graphics: Scatter Plots (see also J5a2)

**NAG**

**GO1AGF** Performs a scatter plot of two variables on a character printing device, with a chosen number of character positions in each direction.

**SAS (IBM 4341)**

**PLOT** Graphs one variable against another, producing a printer plot. The coordinates of each point on the plot correspond to the two variables' values in one or more observations of the input data set. Options to specify the plotting character to be used (plotting character may be the value of another variable), overlay plots, and contour plots are available. Additional options documented in *SAS User's Guide: Basics - Version 5*.

**STATLIB**

**PLT** Is a line-printer plotting routine which produces a simple scatter plot of the N data points whose y (vertical) and x (horizontal) coordinates are contained in N rows of the X and Y vectors. The scale limits are automatically chosen from the data. The plot size is one full page (51 by 101 plotting positions) of line-printer paper.

**PLTH** Is a line-printer plotting routine identical to STATLIB routine PLT, except that the plot size is a half page (51 by 51 plotting positions) of line-printer paper, which can be reproduced onto 8-1/2 X 11 paper without reduction.

**PLTHL** Is a line-printer plotting routine identical to STATLIB routine PLTH, except that the user must specify upper and lower bounds for both x and y axes.

**PLTL** Is a line-printer plotting routine identical to STATLIB routine PLT, except that the user must specify upper and lower bounds for both x and y axes.

L03c. Statistical Graphics: Symbol Plots

**SAS (IBM 4341)**

**PLOT** Graphs one variable against another, producing a printer plot. The coordinates of each point on the plot correspond to the two variables' values in one or more observations of the input data set. Options to specify the plotting character to be used (plotting character may be the value of another variable), overlay plots, and contour plots are available. Additional options documented in *SAS User's Guide: Basics - Version 5*.
**STATLIB**

**MPLT**
Is a line-printer plotting routine for producing on a single plot background the scatter plots of several $Y$ vectors versus a single $X$ vector. The scatter plots of up to 25 different $Y$ vectors are labeled by different symbols. The scale limits are automatically set, and the plot size is one full page (51 by 101 plotting positions) of line-printer paper.

**MPLTH**
Is a line-printer plotting routine identical to STATLIB routine MPLT, except that the plot size is half of one full page (51 by 51 plotting positions) of line printer paper, which can be reproduced onto 8-1/2 X 11 paper without reduction.

**MPLTHL**
Is a line-printer plotting routine identical to STATLIB routine MPLTH, except that the user must specify upper and lower bounds for both the $X$ and $Y$ axes.

**MPLTL**
Is a line-printer plotting routine identical to STATLIB routine MPLT, except that the user must specify upper and lower bounds for the $X$ and $Y$ axes.

**SPLT**
Is a line-printer plotting routine identical to STATLIB routine PLT, except that the user must specify the plotting symbol for each data point.

**SPLTH**
Is a line-printer plotting routine identical to STATLIB routine PLTH, except that the user must specify the plotting symbol for each data point.

**SPLTHL**
Is a line-printer plotting routine identical to STATLIB routine PLTHL, except that the user must specify the plotting symbol for each data point.

**SPLTL**
Is a line-printer plotting routine identical to STATLIB routine PLTL, except that the user must specify the plotting symbol for each data point.

---

**L03d. Statistical Graphics: Probability Plots**

**IMSL**

**USPRP**
Probability plot.

**NAG**

**G01AHF**
Performs a scatter plot of a vector against the normal scores for a sample of the same size, on a character printing device, with a chosen number of character positions in each direction.

**SAS (IBM 4341)**

**UNIVARIATE**
Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of
the weights, largest value, smallest value, range, mode, and signed rank statistic. Observations may be weighted and missing values can be excluded. Documented in *SAS User's Guide: Basics - Version 5*.

---

**L03e. Statistical Graphics: Exploratory Data Analysis Graphics**

**EDA**

**STMNLF**
Produces a stem-and-leaf display of data on a character printing device.

**BOXES**
On the line printer, print adjacent boxplots on a single scale for a given set of values.

**PLOT**
Plot ordered pairs in a condensed manner on the line printer.

**HAXIS**
Print a horizontal axis for a boxplot or regular plot on the line printer.

**IMSL**

**USBOX**
Print a boxplot \(k\) samples.

**USSLF**
Print a stem and leaf display.

**SAS (IBM 4341)**

**SPLIT**
Produces side-by-side schematic printer plots for several groups of observations, such as those described by Tukey (1970). Documented in *SUGI Supplemental Library User's Guide - 1988 Edition*.

**UNIVARIATE**

Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and-leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of the weights, largest value, smallest value, range, mode, and signed rank statistic. Observations may be weighted and missing values can be excluded. Documented in *SAS User's Guide: Basics - Version 5*. 
**L03f. Statistical Graphics: Time Series Plots**

*TIMEPLOT*

Used to plot one or more variables over time intervals. For each plot requested, PROC TIMEPLOT produces a plot where the vertical axis represents the sequence of observations in the data set and the horizontal axis represents values of the variable being examined. A plot produced by TIMEPLOT may occupy more than one page, each observation appears sequentially on a separate line of the plot, and each observation in the plot is accompanied by a printout of the values plotted. PROC TIMEPLOT can be used for combining multiple observations into a single line in the plot and overlaying multiple plots on a single set of axes. Documented in *SAS User’s Guide: Basics - Version 5*.

*STATLIB*

**VPLT**

Provides line-printer plotting of \( N \) observations in vector \( Y \) (usually a time series) versus order (1 to \( N \)). The \( y \) axis is horizontal, and the scale is automatically determined from the range of data. The order scale is vertical, with one observation per printer line. The plot runs over as many pages as necessary (60 points per page). The user may select the density of points plotted. The sample average of the series is shown as a vertical line on the plot, and the value of each point is printed in the right margin.

**VPLT2**

Is a line-printer plotting routine identical to STATLIB routine VPLT, except that two time series of equal length are plotted. The user may choose one or two scales but limits will be set from the range of the series (individually or combined). The average of each series is shown as a vertical bar.

**VPLT2L**

Is a line-printer plotting routine identical to STATLIB routine VPLT2 except that the user must specify the scale limits.

**VPLTB**

Is a line-printer plotting routine identical to STATLIB routine VPLT except that each point is connected to the average line by a bar.

**VPLTBL**

Is a line-printer plotting routine identical to STATLIB routine VPLTB except that the user must specify the scale limits.

**VPLTL**

Is a line-printer plotting routine identical to STATLIB routine VPLT except that the user must specify the scale limits.
L03g. Statistical Graphics: Other Techniques

**IMSL**

**USPC** Print a sample probability density function, a theoretical pdf and confidence band information. Plot these on option.

**USPDF** Plot of two sample probability distribution functions against their spectra.

**SAS (IBM 4341)**

**CHART** Produces vertical and horizontal bar charts, block charts, pie charts, and star charts. Options include making each bar or section represent the percentage of observations, the frequency of observations, the cumulative percentage of observations, the sum of observations by another variable or the mean of observations by another variable. Additional options documented in *SAS User's Guide: Basics - Version 5*.

**PLOT** Graphs one variable against another, producing a printer plot. The coordinates of each point on the plot correspond to the two variables' values in one or more observations of the input data set. Options to specify the plotting character to be used (plotting character may be the value of another variable), overlay plots, and contour plots are available. Additional options documented in *SAS User's Guide: Basics - Version 5*.


L04a. Confidence Intervals, Hypothesis Testing

**IMSL**

**BELBIN** Interval estimate of the parameter \( p \) of the binomial distribution.

**BELPOS** Interval estimate of the parameter lambda of the Poisson distribution.

**BEMNON** Location (mean) inferences using a sample from a normal population with known variance.

**BEMSON** Mean and variance inferences using a sample.

**BENSON** Variance inferences using a sample from a normal population with known mean.

**BEPAT** Mean and variance inferences using samples from each of two normal populations with unequal variances.

**BEPET** Mean and variance inferences using samples from each of two normal populations with equal variances.
BESTA2
Computations of confidence intervals and other basic statistics using output from IMSL routine BESTAT.

GTNOR
Test for normality of random deviates.

OIND
Wilks test for the independence of $k$ sets of multi-normal variates.

OTMLNR
Maximum likelihood estimation from grouped and/or censored normal data.

**SAS (IBM 4341)**

KSLTEST

MEANS
Produces simple univariate descriptive statistics for numeric variables, including number of observations, number of missing values, mean, standard deviation, smallest value, largest value, range, sum, variance, uncorrected sum of squares, corrected sum of squares, standard error of the mean, coefficient of variation, measure of skewness, measure of kurtosis, Student's $t$ value for testing the hypothesis that the population mean is zero, the probability of a greater absolute value of Student's $t$, and the sum of the WEIGHT variable values. Observations may be weighted, printing may be suppressed, and selected statistics may be output to a SAS data set. Missing values are excluded before calculating statistics. Documented in *SAS User's Guide: Basics - Version 5*.

PAIRED
Computes sample statistics on pairs of numeric variables and their differences. The procedure also performs a paired $t$ test, a test of normality of the differences, a sign test, and a signed rank test. Documented in “Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131”.

SUMMARY
Computes descriptive statistics on numeric variables in a SAS data set and outputs the results to a new SAS data set. By using the CLASS statement the user can name the variables to be used to form subgroups, so statistics can be calculated for each subgroup as well as for the whole sample. Statistics computed can include the number of observations in the subgroup with nonmissing values, number of observations in the subgroup having missing values for the variable, as well as all the statistics produced by PROC MEANS. Missing values can be ignored or treated as valid observations. Observations may be weighted. Documented in *SAS User’s Guide: Basics - Version 5*.

TTEST
Computes a $t$ statistic for testing the hypothesis that the means of two groups of observations in a SAS data set are equal. Documented in *SAS User’s Guide: Statistics - Version 5*.

UNIVARIATE
Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and-leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of the weights, largest value, smallest value, range, mode, and signed rank statistic. Observations may be weighted and missing values can be excluded. Documented in *SAS User’s Guide: Basics - Version 5*.

---

File Management

230
NCAR Software Catalog

Nonparametric (Distribution-Free) Analysis (see also L07c, L11a, L11b)

IMSL

NAK1  Kruskal-Wallis test for identical populations.
NBCYC  Noether's test for cyclical trend.
NBQT  Cochran q test.
NBSDL  Cox and Stuart sign test for trends in dispersion and location.
NBSIGN  Sign test (for percentiles).
NDKER  Nonparametric probability density function (one dimensional) estimation by the kernel method.
NDMPLE  Nonparametric probability density function (one dimensional) estimation.
NHEXT  Fisher's exact method for 2 by 2 tables.
NHINC  Includance test.
NK51  Kolmogorov-Smirnov one-sample test.
NK52  Kolmogorov-Smirnov two-sample test.
NMCC  Calculate and test the significance of the Kendall coefficient of concordance.
NMKSF  Frequency distribution of $k$ and the probability of equalling or exceeding $k$, where $k$, the total score from the Kendall rank correlation coefficient calculations, and $n$, the sample size, are given.
NMKTS  $k$-sample trends test against ordered alternatives.
NRBHA  Bhapkar $v$ test.
NRWMD  Wilcoxon signed rank test.
NRWRST  Wilcoxon rank-sum test.

NAG

G08AAF  Performs the Sign test on two related samples of size $N$.
G08ABF  Performs the Wilcoxon matched pairs signed ranks test on two related samples of size $N$.
G08ACF  Performs the Median test on two independent samples of possibly unequal size.
G08ADF  Performs the Mann-Whitney $U$ test on two independent samples of possibly unequal size.
G08BAF  Performs Mood's and David's tests for dispersion differences between two independent samples of possibly unequal size.
G08CAF  Performs the one sample Kolmogorov-Smirnov distribution test.
G08DAF  Calculates Kendall's coefficient of concordance on $k$ independent rankings of $n$ objects or individuals.

**MRANK**
Performs multivariate rank tests using the method of m-rankings. It may be considered a generalization of Friedman's test and includes as special cases many other common nonparametric tests, such as the Kruskal-Wallis and Mann-Whitney tests. MRANK can analyze unbalanced factorial designs with any number of factors and covariates and can compute nonparametric regression and correlation statistics, including Spearman's rank correlation coefficient. MRANK cannot test hypotheses of additivity or do estimation. Documented in *SUGI Supplemental Library User's Guide - 1983 Edition*.

**PAIRED**
Computes sample statistics on pairs of numeric variables and their differences. The procedure also performs a paired t test, a test of normality of the differences, a sign test, and a signed rank test. Documented in "Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131".

**RANK**
Computes ranks for one or more numeric variables across the observations of a SAS data set. The ranks are output to a new SAS data set. Alternatively, PROC RANK produces normal scores or other rank scores. Documented in *SAS User’s Guide: Statistics - Version 5*.

**L04c. Goodness-of-Fit Tests (see also L09)**

**IMSL**

**GFIT**
Chi-squared goodness of fit test.

**GTON**
Sample size or number of class intervals determination for chi-squared test applications.

**GTNOR**
Test for normality of random deviates.

**SAS (IBM 4341)**

**KSLTEST**

**PAIRED**
Computes sample statistics on pairs of numeric variables and their differences. The procedure also performs a paired t test, a test of normality of the differences, a sign test, and a signed rank test. Documented in "Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131". 
UNIVARIATE

Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and-leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of the weights, largest value, smallest value, range, mode, and signed rank statistic. Observations may be weighted and missing values can be excluded. Documented in *SAS User’s Guide: Basics - Version 5.*

L05a. Statistical Distributions: Beta

**AMOSLIB**

BETBIC Computes an *n* member sequence of beta distributions \( y(k) = I_x(a, b + k - 1) \), \( k = 1, \ldots, n \), \( a > 0 \), \( b > 0 \), and \( 0 < x < 1 \), where \( I_x(a, b) \) is the incomplete beta function normalized to 1.

**IMSL**

MDBETA Beta probability distribution function.

MDBETI Inverse beta probability distribution function.

**NAG**

G01BDF Returns the probability, \( I_x(a, b) \) associated with the lower tail of the Beta distribution of the first kind with parameters \( a \) and \( b \).

G01CDF Returns the deviate, \( x(p) \), associated with the given lower tail probability \( p \) of the Beta distribution of the first kind with parameters \( a \) and \( b \), through the function name.

**SAS (IBM 4341)**


233 File Management
L05b. Statistical Distributions: Binomial

**IMSL**

**MDBIN**

Binomial probability distribution function.

**SAS (IBM 4341)**

**PROBNML**


---

L05d. Statistical Distributions: Chi-squared

**AMOSLIB**

**FCHISQ**

Computes the cumulative chi-square distribution \( f(x) \) or its complement \( 1 - f(x) \) with \( n \) degrees of freedom, \( x \geq 0 \) and \( n \geq 1 \).

**FNCHI2**

Computes cumulative non-central chi-square distribution.

**IMSL**

**MDCH**

Chi-squared probability distribution function.

**MDCHI**

Inverse chi-squared probability distribution function.

**MDCHN**

Non-central chi-squared probability distribution function.

**NAG**

**G01BCF**

Returns the probability, \( P(x, n) \), associated with the upper tail of the chi-square distribution with \( n \) degrees of freedom, through the function name.

**G01CCF**

Returns the deviate, \( x(p) \), associated with the given lower tail probability \( p \) of the chi-square distribution with \( n \) degrees of freedom, through the function name.

**SAS (IBM 4341)**

**CINV**

A function which is the inverse of CPROB, the value \( X \) exceeded with probability \( 1 - P \) by a chi-square random variable with \( DF \) degrees of freedom and noncentrality parameter \( NC \). Documented in *SUGI Supplemental Library User's Guide - 1988 Edition*.

**CNOCT**

A function which returns the noncentrality parameter such that the probability of a noncentral chi-square random variable with degrees of freedom \( DF \) exceeding \( X \) is \( 1 - P \). Documented in *SUGI Supplemental Library User's Guide - 1988 Edition*. 

---

File Management
A function which computes the probability that a random variable distributed as chi-square with \( DF \) degrees of freedom and noncentrality parameter \( NC \) falls below the value \( X \). Documented in *SUGI Supplemental Library User's Guide - 1983 Edition*.


**L05e. Statistical Distributions: Circular Coverage Function**

*AMOSLIB*

**FCIRCVC** The circular coverage function gives the probability of covering a point target by a weapon of radius \( a \) when the offset aim is \( d \), and the distribution of aiming errors is circular normal with standard deviation \( \sigma \).

**L05i. Statistical Distributions: F Distribution**

*AMOSLIB*

**FFDIST** Computes the cumulative \( f \) distribution \( f(x) \) or its complement \( 1 - f(x) \) for a random variable \( f = \frac{u/m}{v/n} \) where \( u \) is chi-square(\( m \)) and \( v \) is chi-square(\( n \)).

**FMLTVF** Computes the joint cumulative distribution for the \( F \)-random variables \( F(k) = \frac{u(k)/F(m(k))}{v/F(n)} \) where each \( u(k) \) is chi-square with \( F(m(k)) \) degrees of freedom and \( v \) is chi-square with \( F(n) \) degrees of freedom, \( k = 1, \ldots, nn \) and each \( u(k) \) and \( v \) are independent.

**FNCF** Computes cumulative noncentral \( f \) distribution by recursion.

**FNCFQ** Computes cumulative noncentral \( f \) distribution by quadrature.

*IMSL*

**MDFD** \( f \) probability distribution function.

**MDFDRE** \( f \) probability distribution function (integer or fractional degrees of freedom).

**MDFI** Inverse \( f \) probability distribution function.
NCAR Software Catalog

**NAG**

**GO1BBF**  
Returns the probability associated with the upper tail of the $F$ or variance-ratio distribution with $m$ and $n$ degrees of freedom.

**GO1CBF**  
Returns the deviate, $x(p)$ associated with the lower tail probability $p$ of the $F$ variance-ratio distribution with $m$ and $n$ degrees of freedom, through the function name.

**SAS (IBM 4341)**

**FINV**  
A function which is the inverse of FPROB, the value $X$ exceeded with probability $1-P$ by an $F$ random variable with $DF_1$ degrees of freedom in the numerator and $DF_2$ in the denominator, and with noncentrality parameter NC. Documented in *SUGI Supplemental Library User's Guide - 1983 Edition*.

**FNONCT**  
A function which returns the noncentrality parameter such that the probability of a noncentral $F$ random variable with degrees of freedom $DF_1$ in the numerator and $DF_2$ in the denominator exceeding $X$ is $1-P$. Documented in *SUGI Supplemental Library User's Guide - 1983 Edition*.

**FPROB**  
A function which computes the probability that a random variable distributed as $F$ with $DF_1$ degrees of freedom in the numerator and $DF_2$ in the denominator and with noncentrality parameter NC falls below the value $X$. Documented in *SUGI Supplemental Library User's Guide - 1983 Edition*.

**PROBF**  

**L05j. Statistical Distributions: Gamma**

**IMSL**

**MDGAM**  
Gamma probability distribution function.

**SAS (IBM 4341)**

**GAMINV**  

**PROBGAM**  
Function which computes the probability values for the gamma distribution. Documented in *SAS User's Guide: Basics - Version 5*.
L05m. Statistical Distributions: Hypergeometric

**IMSL**

MDHYP Hypergeometric probability distribution function.

**SAS (IBM 4341)**


L05n. Statistical Distributions: Kolmogorov-Smirnov

**IMSL**

MDSMR Kolmogorov-Smirnov statistics asymptotic probability distribution function.

L05r. Statistical Distributions: Negative Binomial

**SAS (IBM 4341)**

L05s. Statistical Distributions: Normal

**AMOSLIB**

FNORM  Computes the cumulative normal distribution $f(x)$ or its complement $1-f(x)$.

FNORMB  Computes the cumulative bivariate normal probability
$p(r_1 \leq x_1, r_2 \leq x_2, \rho)=n(x_1, x_2, \rho)$ for random variables $r_1$ and $r_2$ with correlation $\rho$.

**IMSL**

MDBNOR  Bivariate normal probability distribution function.

MDNOR  Normal or Gaussian probability distribution function.

MDNRIS  Inverse standard normal (Gaussian) probability distribution function.

MSENO  Expected values of normal order statistics.

MSMRAT  Ratio of the ordinate to the upper tail area of the standardized normal (Gaussian) distribution.

**NAG**

G01CEF  Returns the deviate, $x(p)$, associated with the given lower tail probability $p$ of the standardised normal distribution, through the function name.

G01DAF  Computes a set of normal scores, i.e. the expected values of an ordered set of independent observations from a normal distribution with mean 0.0 and standard deviation 1.0.

S15ABF  Returns the value of the cumulative normal distribution function, $P(x)$.

S15ACF  Returns the value of the complement of the cumulative normal distribution function, $Q(x)$.

**SAS (IBM 4341)**


L05u. Statistical Distributions: Poisson

**IMSL**

**MDTPS** Cumulative probability and, optionally, individual terms of the Poisson probability distribution function.

*SAS (IBM 4341)*


L05v. Statistical Distributions: T Distribution

**AMOSLIB**

**FBIVTS** Computes bivariate $t$ distribution by means of a series.

**FCENT** Computes the cumulative $t$ distribution $f(x)$ or its complement $1-f(x)$ for a random variable $t = \frac{u}{\sqrt{v/n}}$ where $u$ is normal (0,1) and $v$ is chi-square($n$).

**FDNCT** Computes the doubly non-central $t$-distribution.

**FMLTVT** Computes multivariate $t$ with correlation $\rho$ using evaluation of some cumulative distribution functions by numerical quadrature.

**FNCT** Computes cumulative non-central $t$-distribution.

**IMSL**

**MDSTI** Inverse of a modification of Students $t$ probability distribution function.

**MDTD** Students $t$ probability distribution function.

**MDTN** Non-central $t$ probability distribution function.

**MDTNF** Integral related to calculation of noncentral $t$ and bivariate normal probability distribution functions.

**NAG**

**G01BAF** Returns the probability associated with the lower tail of the Student's $t$ distribution with $n$ degrees of freedom.

**G01CAF** Returns the deviate, $x(p)$ associated with the given lower tail probability $p$ of Student's $t$ distribution with $n$ degrees of freedom, through the function name.
PROBT

TINV
A function which is the inverse of TPLOB, the value $X$ exceeded with probability $1-P$ by a Student’s $t$ random variable with $DF$ degrees of freedom and noncentrality parameter $NC$. Documented in SUGI Supplemental Library User’s Guide - 1983 Edition.

TNOCT
A function which returns the noncentrality parameter such that the probability of a noncentral $T$ random variable with $DF$ degrees of freedom exceeding $X$ is $1-P$. Documented in SUGI Supplemental Library User’s Guide - 1983 Edition.

TPROB
A function which computes the probability that a random variable distributed as Student’s $t$ with $DF$ degrees of freedom and noncentrality parameter $NC$ falls below the value $X$. Documented in SUGI Supplemental Library User’s Guide - 1983 Edition.

---

**L05y.** Statistical Distributions: General Distribution

**IMSL**

MDGC
General cumulative probability distribution function, given ordinates of the density.

MDGCI
Inverse of a general cumulative probability distribution function, given ordinates of the density.

NDEST
Evaluate probability density function at specified points.

---

**L06a.** Pseudo-Random Number Generators: Beta

**IMSL**

GBBTR
Beta random deviate generator (rejection method).

**NAG**

G05DLF
Returns a pseudo-random real number taken from a beta distribution of the first kind with parameters $G$ and $H$.

G05DMF
Returns a pseudo-random real number taken from a beta distribution of the second kind with parameters $G$ and $H$.
L06b. Pseudo-Random Number Generators: Binomial

**IMSL**

**GGBN** Binomial random deviate generator.

**NAG**

**G05DFZ** Returns a pseudo-random logical value - TRUE with probability $P$ and FALSE with probability $(1-P)$.

**G05EDF** Sets up the reference vector $R$ for a binomial distribution of the number of successes in $N$ trials, each with probability of success $P$.

**SAS (IBM 4341)**


L06c. Pseudo-Random Number Generators: Cauchy

**IMSL**

**GGCAY** Cauchy random deviate generator.

**NAG**

**G05DFF** Returns a pseudo-random real number taken from a Cauchy distribution with median $A$ and semi-interquartile range $B$.

**SAS (IBM 4341)**

L06d. Pseudo-Random Number Generators: Chi-squared

**IMSL**

GGCHS Chi-squared random deviate generator.

**NAG**

G05DHF Returns a pseudo-random real number taken from a chi-square distribution with $N$ degrees of freedom.

L06g. Pseudo-Random Number Generators: Exponential

**IMSL**

GGEXN Exponential random deviate generator.

GGEXT Random deviate generator for a mixture of two exponentials.

**NAG**

G05DBF Returns a pseudo-random real number taken from a (negative) exponential distribution with mean $A$.

**SAS (IBM 4341)**


L06i. Pseudo-Random Number Generators: F Distribution

**NAG**

G05DKF Returns a pseudo-random real number taken from Snedecor's $F$ (or Fisher's variance ratio) distribution with $M$ and $N$ degrees of freedom.
L06j. Pseudo-Random Number Generators: Gamma

**IMSL**

GGAMR One parameter gamma random deviate generator, and usable as the basis for two parameter gamma, exponential, chi-squared, chi, beta, t, and f deviate generation.

**NAG**

G05DGF Returns a pseudo-random real number taken from a gamma distribution with parameters $G$ and $H$.

**SAS (IBM 4341)**


L06k. Pseudo-Random Number Generators: Geometric

**IMSL**

GGEOT Geometric random deviate generator.

L06m. Pseudo-Random Number Generators: Hypergeometric

**IMSL**

GGHPR Hypergeometric random deviate generator.

**NAG**

G05EFF Sets up a reference vector $R$ for a hypergeometric distribution of the number of specified items in a sample of size $L$, taken from a population of size $N$ with $M$ specified items in it.
L06p. Pseudo-Random Number Generators: Logistic

NAG

G05DCF Returns a pseudo-random real number taken from a logistic distribution with mean $A$ and spread $B$.

L06q. Pseudo-Random Number Generators: Lognormal

IMSL

GGNLG Log-normal random deviate generator.

NAG

G05DEF Returns a pseudo-random real number taken from a lognormal distribution with parameters $A$ and $B$.

L06r. Pseudo-Random Number Generators: Negative Binomial

IMSL

GGBNR Negative binomial random deviate generator.

NAG

G05EEF Sets up the reference vector $R$ for a negative binomial distribution of the number of successes before $N$ failures, where each trial has probability of success $P$. 
L06s. Pseudo-Random Number Generators: Normal

**AMOSLIB**

**RVNORM** Computes a value for a normal random variable on each call.

**IMSL**

**GGNML** Normal or Gaussian random deviate generator.
**GGNO** Generate set of order statistics from normal distribution.
**GGNPM** Normal random deviate generator via the polar method.
**GGNQF** Normal random deviate generator – function form of IMSL routine GGNML.
**GGNSM** Multivariate normal random deviate generator with given covariance matrix.

**LOCLIB**

**RNDEV** Generates independent random deviates with mean 0 and variance 1.

**NAG**

**G05DDF** Returns a pseudo-random real number taken from a normal (Gaussian) distribution with mean $A$ and standard deviation $B$.
**G05EAF** Sets up a reference vector for a multivariate normal distribution.
**G05EZF** Generates a pseudo-random vector from a multivariate normal distribution, using the reference vector set up by G05EAF.

**SAS (IBM 4341)**


**SLATEC**

**RGAUSS** Generates a normally distributed (Gaussian) random number.

**STATLIB**

**NRAND** Generates a vector of normal random numbers having mean zero and standard deviation 1.0. Data having a different mean and standard deviation may be obtained through a linear transformation.
L06u. Pseudo-Random Number Generators: Poisson

**IMSL**

GGNPP Nonhomogeneous Poisson process generator with rate function \( \lambda(t) \) – fixed interval, fixed number, or one at a time.

GGPON Poisson random deviate generator where the Poisson parameter changes frequently.

GGPOS Poisson random deviate generator where the Poisson parameter does not change often.

**NAG**

G05ECF Sets up the reference vector \( R \) for a Poisson distribution with mean \( T \).

**SAS (IBM 4341)**

RANPOI Function which generates an observation from a Poisson distribution with parameter \( \lambda \). Documented in *SAS User’s Guide: Basics - Version 5*.

L06v. Pseudo-Random Number Generators: T Distribution

**NAG**

G05DJF Returns a pseudo-random real number taken from a Student’s \( t \)-distribution with \( N \) degrees of freedom.

L06w. Pseudo-Random Number Generators: Uniform and Associated Uniform Generator Tests

**IMSL**

GGUBFS Basic uniform \((0,1)\) random number generator – function form of IMSL routine GGUBS.

GGUBS Basic uniform \((0,1)\) pseudo-random number generator.

GGUBT Uniform \((0,1)\) pseudo-random number generator using alternate multiplier.
Discrete uniform random number generator.
Generate set of order statistics from uniform (0,1) distribution.
Uniform (0,1) random number generator with shuffling.
The \( d \)-square test.
\( d \)-square tally.
Moments and standardized moments of uniform random numbers.
Count of the number of zero bits in a given subset of a real word.
Probability distribution of \( n \) elements into two equi-probable states.
Poker test tally of hand types and statistics.
The poker test.
Tally of coordinates of pairs (or lagged pairs) of random numbers.
Pairs test or goods serial test.
Runs test.
Tally of number of runs up and down.
Tally for triplets test.
Triplets test.

**NAG**

**G05CAF** Returns a pseudo-random number taken from a uniform distribution between 0 and 1.
**G05CBF** Sets the basic generator routine G05CAF to a repeatable initial state.
**G05CCF** Sets the basic generator routine G05CAF to a non-repeatable initial state.
**G05CFF** Saves the current state of the basic generator routine G05CAF.
**G05CGF** Restores the state of the basic generator routine G05CGF after a previous call to G05CFF.
**G05DAF** Returns a pseudo-random real number taken from a uniform distribution between \( A \) and \( B \).
**G05DYF** Returns a pseudo-random integer taken from a uniform distribution between \( M \) and \( N \) (inclusive).
**G05EBF** Sets up the reference vector \( R \) for a discrete uniform distribution between \( M \) and \( N \) inclusive.

**PORT**

**RANBYT** Returns the real random variate generated by the PORT routine, UNI, together with its bit pattern presented in four 8-bit bytes. The exact bit patterns of the uniform random variates may be needed in order to implement, portably, random generators employing algorithms, such as rejection techniques, requiring exact boundary values.
**RANSET** Initializes the uniform random number generator, UNI, to other than the default initial values.
Returns a single real random variate from the uniform \([0,1)\) distribution. This will produce the same sequence of values, to the accuracy of the computer, on any computer with 16 or more bits per word.

**SAS (IBM 4841)**

**RANUNI**

**UNIFORM**
Function which generates a pseudo-random variate uniformly distributed on the interval \((0,1)\). Documented in *SAS User's Guide: Basics - Version 5.*

**SLATEC**

**RAND**
Generates a uniformly distributed random number.

**RUNIF**
A portable random number generator.

**L06x.** Pseudo-Random Number Generators: Weibull

**IMSL**

**GGWIB**
Weibull random deviate generator.

**NAG**

**G05DPF**
Returns a pseudo-random number taken from a two parameter Weibull distribution with shape parameter \(A\) and scale parameter \(B\).

**L06y.** Pseudo-Random Number Generators: General Distribution

**IMSL**

**GGDA**
General discrete distribution random deviate generator using alias method.

**GGDT**
General discrete distribution random deviate generator using table lookup method.

**GGVCR**
General continuous distribution random deviate generator.
NAG

G05EXF  Sets up the reference vector $R$ for a discrete distribution with PDF (probability density function) or CDF (cumulative distribution function) $P$.

G05EYF  Returns a pseudo-random integer taken from a discrete distribution defined by a reference vector $R$.

SAS (IBM 4341)


L06z. Pseudo-Random Number Generators: Other Distributions

IMSL

GGCOR   Generate a random orthogonal matrix and a random correlation matrix.

GGMTN   Multinomial random deviate generator.

GGPER   Generate a random permutation of the integers 1 to $k$.

GGSPH   Generation of uniform random deviates from the surface of the unit sphere in 3 or 4 space.

GGSRS   Generate a simple random sample from a finite population.

GGSTA   Stable distribution random deviate generator.

GGTAB   Generate a random contingency table with given row and column totals.

GGTRA   Triangular distribution random deviate generator.

GGVMS   Von Mises random deviate generator.

NAG

G05DZF   Returns a pseudo-random logical value - TRUE with probability $P$ and FALSE with probability $(1-P)$.

G05EGF   Sets up a reference vector for an autoregressive moving-average (ARMA) time series model with normally distributed errors, so that G05EWF may be used to generate successive terms. It also initialises the series to a stationary position.

G05EHF   Generates a pseudo-random permutation of a vector.

G05EJF   Generates a pseudo-random subset of a vector.
SAS (IBM 4341)

**RANTRI**  

---

L07a.  
**Generation of Experimental Designs**

**IMSL**

**RLCOMP**  
Generation of an orthogonal central composite design.

**SAS (IBM 4341)**

**PLAN**  
Generates randomized plans for experiments, including completely randomized designs, split-plot designs, and hierarchical designs. These plans are represented as groups of random permutations of positive integers. Documented in *SAS User's Guide: Statistics - Version 5.*

---

L07b.  
**Analysis of Variance**

**IMSL**

**ABIBN**  
Analysis of balanced incomplete block and balanced lattice designs.

**ACRDAN**  
Analysis of one-way classification design data.

**AFACN**  
Full factorial plan analysis.

**AFACT**  
Sums of squares, mean squares, degrees of freedom, and means for all effects in a full factorial plan, allowing replication on option.

**AGBACP**  
Analysis of balanced complete experimental design structure data.

**AGLMOD**  
General linear model analysis.

**ALSQAN**  
Analysis of Latin square design data.

**ANESTE**  
Analysis of completely nested design data with equal numbers in the subclasses.

**ANESTU**  
Analysis of completely nested design data with unequal numbers in the subclasses.

**ARCBAN**  
Analysis of two-way classification design data.
NAG

G04ADF Performs the analysis of variance for a Latin square design.

G04AEF Performs an analysis of variance for a one-way classification with treatment groups of possibly unequal size, and also computes the treatment group means.

G04AFF Performs an analysis of variance for a two-way cross-classification with equal cell frequencies, and computes the treatment group means.

G04AGF Performs an analysis of variance for a two-way hierarchical classification with subgroups of possibly unequal size, and also computes the treatment group and subgroup means.

SAS (IBM 4341)

ANOVA Performs univariate and multivariate analysis of variance for balanced data (data with equal numbers of observations for every combination of the classification factors) and, in addition will handle Latin-square designs, certain balanced incomplete blocks designs, completely nested (hierarchical) designs, and designs whose cell frequencies are proportional to each other. Many tests are provided to compare means. Observations may be weighted by an integer value. Documented in SAS User's Guide: Statistics - Version 5.

GLM Uses the method of least squares to fit general linear models. GLM handles classification variables, which have discrete levels, as well as continuous variables, which measure quantities. Thus GLM can be used for many different analyses including: simple regression, multiple regression, analysis of variance (especially for unbalanced data), analysis of covariance, response-surface models, weighted regression, polynomial regression, partial correlation, multivariate analysis of variance, and repeated measures analysis of variance. GLM can provide tests of hypotheses for the effects of a linear model regardless of the number of missing cells or the extent of confounding, the form of the estimable functions employed in the test, and the general form of all estimable functions. When one or more effects in the model are random, GLM prints the coefficients of the expected mean squares, computed correctly even if there are missing cells by using the synthesis method. Additional options are documented in the SAS User's Guide: Statistics - Version 5.


251 File Management Category L07b
STATLIB

ONEWY  Provides one-way analysis of variance of two or more random samples with arbitrary (perhaps unequal) sample sizes. Printed output includes the usual analysis of variance table, the robust Kruskal-Wallis rank test, comprehensive summary statistics, and pairwise multiple comparisons using the following techniques: Newman-Keuls and Scheffe, Cochran’s $C$, Bartlett-Box and variance ratio tests for homogeneity of variances, and the random effects model components of variance estimate.

ONEWYS  Is a one-way analysis of variance routine for two or more random samples with arbitrary (perhaps unequal) sample sizes. It is identical to STATLIB routine ONEWY except that it provides storage of computed statistics and has optional printout.

L07c.  Nonparametric Analysis of Variance

IMSL

NAWRP  Wilsons ANOVA (2 or 3 way designs) without replicates.
NAWRPE  Wilsons ANOVA (1, 2, or 3 way designs) with equal replication.
NAWRPU  Wilsons ANOVA (1, 2, or 3 way designs) with unequal replication.

NAG

G08AEF  Performs the Friedman two-way analysis of variance by ranks on $K$ related samples of size $N$.
G08AFF  Performs the Kruskal-Wallis one-way analysis of variance by ranks on $k$ independent samples of possibly unequal sizes.

SAS (IBM 4341)

NPAR1WAY  Performs analysis of variance on ranks and certain rank scores of a response variable across a one-way classifications. NPAR1WAY is a nonparametric procedure for testing that the distribution of a variables has the same location parameter across different groups. Documented in SAS User’s Guide: Statistics - Version 5.
L07d. Analysis of Covariance

IMSL

ANCOV1 Covariance analysis for one-way classification design data.

SAS (IBM 4341)

ANOVA Performs univariate and multivariate analysis of variance for balanced data (data with equal numbers of observations for every combination of the classification factors) and, in addition will handle Latin-square designs, certain balanced incomplete blocks designs, completely nested (hierarchical) designs, and designs whose cell frequencies are proportional to each other. Many tests are provided to compare means. Observations may be weighted by an integer value. Documented in SAS User's Guide: Statistics - Version 5.

GLM Uses the method of least squares to fit general linear models. GLM handles classification variables, which have discrete levels, as well as continuous variables, which measure quantities. Thus GLM can be used for many different analyses including: simple regression, multiple regression, analysis of variance (especially for unbalanced data), analysis of covariance, response-surface models, weighted regression, polynomial regression, partial correlation, multivariate analysis of variance, and repeated measures analysis of variance. GLM can provide tests of hypotheses for the effects of a linear model regardless of the number of missing cells or the extent of confounding, the form of the estimable functions employed in the test, and the general form of all estimable functions. When one or more effects in the model are random, GLM prints the coefficients of the expected mean squares, computed correctly even if there are missing cells by using the synthesis method. Additional options are documented in the SAS User's Guide: Statistics - Version 5.


L07e. Multivariate Analysis of Variance

SAS (IBM 4341)

ANOVA Performs univariate and multivariate analysis of variance for balanced data (data with equal numbers of observations for every combination of the classification factors) and, in addition will handle Latin-square designs, certain balanced incomplete blocks designs, completely nested (hierarchical) designs, and designs whose cell frequencies are proportional to each other. Many tests are provided to compare means. Observations may be weighted by an integer value. Documented in SAS User's Guide: Statistics - Version 5.
NCAR Software Catalog

GLM

Uses the method of least squares to fit general linear models. GLM handles classification variables, which have discrete levels, as well as continuous variables, which measure quantities. Thus GLM can be used for many different analyses including: simple regression, multiple regression, analysis of variance (especially for unbalanced data), analysis of covariance, response-surface models, weighted regression, polynomial regression, partial correlation, multivariate analysis of variance, and repeated measures analysis of variance. GLM can provide tests of hypotheses for the effects of a linear model regardless of the number of missing cells or the extent of confounding, the form of the estimable functions employed in the test, and the general form of all estimable functions. When one or more effects in the model are random, GLM prints the coefficients of the expected mean squares, computed correctly even if there are missing cells by using the synthesis method. Additional options are documented in the *SAS User's Guide: Statistics - Version 5.*

L07f. Tests to use with Analysis of Variance and Covariance

**IMSL**

ACTRST Contrast estimates and sums of squares.
AGVACL One or two-sided interval estimate of a variance component.
AGXPM Expected mean squares for balanced complete design models.
AMEANS Preparation of a set of unbalanced data for analysis by the method of unweighted means.
AORDR Reordering of the data obtained from any balanced complete experimental design.
ASNKMC Student-Newman-Keuls multiple comparison test.
NAFRE Friedmans test for randomized complete block designs.

**SAS (IBM 4341)**

L08a1. Regression: Simple Linear

IMSL

BEMIRI Estimates of means, simple regression coefficients, their intercepts, standard errors of the regression coefficients, and standard deviations for arrays which contain missing values. (in-core version)

BEMIRO Estimates of means, simple regression coefficients, their intercepts, standard errors of the regression coefficients, and standard deviations for arrays which contain missing values. (out-of-core version)

NAG

G02CAF Performs a simple linear regression with dependent variable \( Y \) and independent variable \( X \).

G02CBF Performs a simple linear regression with no constant with dependent variable \( Y \) and independent variable \( X \).

G02CCF Performs a simple linear regression with dependent variable \( Y \) and independent variable \( X \), omitting cases involving missing values.

G02CDF Performs a simple linear regression with no constant, with dependent variable \( Y \) and independent variable \( X \), omitting cases involving missing values.

L08a2. Regression: Polynomial

IMSL

RLDCVA Variance estimates for decoded orthogonal polynomial regression coefficients.

RLDCW Variances of coded orthogonal polynomial regression coefficients. For usage in conjunction with IMSL routines RLFOTH and RLFOTW, and provided to prepare input for IMSL routine RLDCVA.

RLDOPM Coefficient decoder for an orthogonal polynomial regression model.

RLFOR Fit a univariate curvilinear regression model using orthogonal polynomials with optional weighting - easy to use version.

RLFOTH Fit a univariate curvilinear regression model using orthogonal polynomials.

RLFOTW Fit a univariate curvilinear regression model using orthogonal polynomials with weighting independent variable settings. The \( z \) vector must not be constant.
**STATLIB**

**PFIT**
Provides linear least squares analysis for fitting a polynomial model whose design matrix is completely user-specified. Automatic printout includes a list of the $N$ observations, four line-printer plots of residuals, and a list of the coefficients plus their associated variance-covariance matrix and basic statistics.

**PFITS**
Provides linear least squares analysis for fitting a polynomial model. This routine is the same as STATLIB routine PFIT, except that it allows the user to store computed results.

**PFITW**
Provides linear least squares analysis for fitting a polynomial model. This routine is the same as STATLIB routine PFIT, except that it allows the user to specify weights.

**PFITWS**
Provides linear least squares analysis for fitting a polynomial model. This routine is the same as STATLIB routine PFIT, except that it allows the user to store computed results and to specify weights.

**L08a3. Regression: Multiple Linear**

**IMSL**

**RLEAP**
Leaps and bounds algorithm for determining a number of best regression subsets from a full regression model.

**RLLMV**
Perform linear regression using the minimax criterion.

**RLMUL**
Multiple linear regression analysis.

**RLSEP**
Selection of a regression model using a forward stepwise algorithm, and computation of the usual analysis of variance table entries – easy to use version.

**RLSTP**
Regression model selection using a forward stepwise algorithm with results available after each step.

**RLSUBM**
Retrieval of a symmetric submatrix from a matrix stored in symmetric storage mode by RLSTP.

**RLSUM**
Reordering of the rows and corresponding columns of a symmetric matrix stored in symmetric storage mode.

**USLEAP**
Print results of the best-regressions analysis performed by IMSL routine RLEAP.

**LOCLIB**

**LSPACK**
Solves both unconstrained and constrained least-squares problems and computes the associated covariance matrices. These routines are from Hanson and Lawson's book *Solving Least-Squares Problems*.
NCAR Software Catalog

NAG

G02CGF Performs a multiple linear regression on a set of variables whose means, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients are given.

G02CHF Performs a multiple linear regression with no constant on a set of variables whose sums of squares and cross-products about zero and correlation-like coefficients are given.

G02CJF Performs one or more multiple linear regressions, regressing each of a set of dependent variables separately on the same set of independent variables. Input to the routine is in the form of raw data. Output includes, for each dependent variable, estimates of the regression coefficients, and an estimate of the variance of residuals.

NCARLB

RGRSN1 Calculates a set of regression coefficients for an unweighted regression model.

RGRSN2 Weights the model with a diagonal matrix of weights.

RGRSN3 Weights the model with the covariance matrix of the random variables.

RGRSN4 Similar to RGRSN3 except that the covariance matrix is banded.

RGRSN5 Weights the model using standard deviations of repeated observations.

RGRSN6 Weights the model using an estimated covariance matrix from repeated observations.

SAS (IBM 4341)


REG Fits least-squares estimates to linear regression models which may have linear constraints. Can handle more than one dependent variable. Prints predicted values, residuals, studentized residuals, and confidence limits, and can output these items to an output SAS data set. Also prints influence and collinearity diagnostics, and produces partial regression leverage plots. Options documented in SAS User’s Guide: Statistics - Version 5.

RSQUARE Selects optimal subsets of independent variables in a multiple regression analysis using the multiple correlation coefficient squared. Regression coefficients and a variety of statistics useful for model selection can be printed or output to a SAS data set. Documented in SAS User’s Guide: Statistics - Version 5.


File Management

257
STATLIB

FIT Provides linear least squares analysis for fitting a general linear model whose design matrix is completely user-specified. Automatic printout includes a list of the \( N \) observations, four line-printer plots of residuals, and a list of the coefficients plus their associated variance-covariance matrix and basic statistics.

FITS Is a linear least squares routine for fitting a general linear model. It is the same as STATLIB routine FIT, except that it allows storage of computed results.

FITW Is a linear least squares routine for fitting a general linear model. It is the same as STATLIB routine FIT, except that it allows the user to specify weights.

FITWS Is a linear least squares routine for fitting a general linear model. It is the same as STATLIB routine FIT, except that it allows the user to store computed results and to specify weights.

L08a4. Regression: Robust

EDA

RLINE For given data, fit a straight line by the "resistant line" technique.

IMSL

RLLAV Perform linear regression using the least absolute values criterion.

SAS (IBM 4341)

LAV Uses the least absolute values criterion to fit a linear model. The model can be of less than full rank, and inclusion of an intercept can be suppressed. Documented in SUGI Supplemental Library User’s Guide - 1988 Edition.
L08b. Regression: Nonlinear

**IMSL**

RSMITZ Least squares fit of the non-linear regression model $y_i = \alpha + \beta x_i + \epsilon_i$.

**NAG**

E04YCF Returns estimates of elements of the variance-covariance matrix of the estimated regression coefficients for a non-linear least squares problem. The estimates are derived from the Jacobian of the function $f(x)$ at the solution.

F04YAF Returns elements of the estimated variance-covariance matrix of the sample regression coefficients for a linear least squares problem.

**SAS (IBM 4341)**

NLIN Produces least-squares or weighted least-squares estimates of the parameters of a nonlinear model using one of four iterative methods. Partial derivatives of the model with respect to each parameter must be specified except for the multivariate secant (DUD) method. Bounds may be put on the parameter estimates and you can define your own objective function to be minimized. NLIN can be used for segmented models and to compute maximum likelihood estimates for certain models. Documented in *SAS User’s Guide: Statistics - Version 5*.

**STAR PAC**

DCKLSC Checks derivatives; prints report; and returns maximal results.

DCKLSC Checks derivatives; user supplies control values; and it returns maximal results.

NLS Performs unweighted nonlinear least squares regression; numerically approximates derivatives; prints five-part report; and returns minimal results.

NLSC Performs unweighted nonlinear least squares regression; numerically approximates derivatives; user-supplies control values; and it returns minimal results.

NLSD Performs unweighted nonlinear least squares regression; user-supplies derivatives; it prints five-part report; and returns minimal results.

NLSDC Performs unweighted nonlinear least squares regression; user-supplies derivatives; user-supplies control values; it returns minimal results.

NLSDS Performs unweighted nonlinear least squares regression; user-supplies derivatives; user-supplies control values; it returns maximal results.

NLSS Performs unweighted nonlinear least squares regression; numerically approximates derivatives; user supplies control values; it returns maximal results.

NLSW Performs weighted nonlinear least squares regression; numerically approximates derivatives; prints five-part report; and returns minimal results.

NLSWC Performs weighted nonlinear least squares regression; numerically approximates derivatives; user supplies control values; and it returns minimal results.

NLSWD Performs weighted nonlinear least squares regression; user-supplies derivatives; it prints five-part report; and returns minimal results.
NLSWDC  Performs weighted nonlinear least squares regression; user-supplies derivatives; user-supplies control values; it returns minimal results.

NLSWDS  Performs weighted nonlinear least squares regression; user-supplies derivatives; user-supplies control values; it returns maximal results.

NLSWS  Performs weighted nonlinear least squares regression; numerically approximates derivatives; user-supplies control values; and it returns maximal results.

STPLS  Selects optimum step size for numerically approximating derivatives; prints report; and returns maximal results.

STPLSC  Selects optimum step size for numerically approximating derivatives; user-supplies control values; and it returns maximal results.

STATLIB

NFIT  Is a nonlinear least squares routine for nonlinear models, providing a printout of the basic analysis, using many control parameters for the Marquardt algorithm preset to generally recommended values.

NFITS  Is a nonlinear least squares routine for nonlinear models. It is identical to STATLIB routine NFIT, except that it has the following options: printout suppression, storing of computed results, and adjustment of the control parameters for the Marquardt algorithm.

NFITW  Is a nonlinear least squares routine for nonlinear models. It is identical to STATLIB routine NFIT, with the option of specifying arbitrary weights in the basic analysis.

NFITWS  Is a nonlinear least squares routine for nonlinear models. It is identical to STATLIB routine NFIT, with the following options: arbitrary weight specification, computation storage, control parameter adjustment for the Marquardt algorithm, and printout suppression.

LOCLIB

LSPACK  Solves both unconstrained and constrained least-squares problems and computes the associated covariance matrices. These routines are from Hanson and Lawson's book *Solving Least-Squares Problems*.

SAS (IBM 4341)

AUTOREG  Estimates parameters in regression models when the data are time series and the error term is an autoregressive process. This procedure works with embedded missing values for the independent or dependent variables. Documented in *SAS/ETS User’s Guide - Version 5*.

COXREGR  Performs regressions on one to twenty independent variables on a censored dependent variable using D.R. Cox's life-table regression model. Documented in
Fits the logistic multiple regression model to a single binary dependent variable or to an ordinal dependent variable. The procedure can fit one model or use either a backward elimination procedure or a stepwise technique. Documented in "SUGI Supplemental Library User's Guide - 1988 Edition" and "Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131".

Estimates regression parameters for time series data when the effects of the regressor variables can be distributed across time. PDLREG uses the Almon distributed lag polynomial with or without endpoint restrictions. The procedure can also produce estimates when a finite autoregressive error structure is specified. Documented in "SAS/ETS User's Guide - Version 5."


Fits the parameters of a complete quadratic response surface and then determines critical values to optimize the response with respect to the factors in the model. Documented in "SAS User's Guide: Statistics - Version 5."

Estimates coefficients in an interdependent system of linear equations using any of the following methods: ordinary least squares, k class, minimum expected loss, two-stage least squares, limited-information maximum likelihood, three-stage least squares, seemingly unrelated regressions or joint generalized last squares. Documented in "SAS/ETS User's Guide - Version 5."

Combines iterative minimization methods for nonlinear regression with specialized estimation techniques for simultaneous equation systems to estimate parameters in a simultaneous system of nonlinear equations. Available estimation methods include: nonlinear ordinary least squares, nonlinear seemingly unrelated regression, nonlinear two-stage least squares, nonlinear three-stage least squares, iterated versions of the above that produce simultaneous estimates of the covariance matrix and the parameters. Documented in "SAS/ETS User's Guide - Version 5."

Decoding of a quadratic regression model.

Pure replication error degrees of freedom and sum of squares – in core version.

Pure replication error degrees of freedom and sum of squares – out of core version.

Centering of independent variable settings and generation of centered square and cross product terms – in core version.

Techniques Used with Regression

IMSL

RLDCQM

RLFITI

RLFITO

RLGQMI
RLGQMO  Centering of independent variable settings and generation of uncentered square and cross product terms - out of core version.

RLINCF  Response control using a fitted simple linear regression model.

RLINPF  Inverse prediction using a fitted simple linear regression model.

RLONE  Analysis of a simple linear regression model.

RLOPDC  Response prediction using an orthogonal polynomial regression model.

RLPOL  Generate orthogonal polynomials with the associated constants aa and bb.

RLPRDI  Confidence intervals for the true response and for the average of a set of future observations on the response – in core version.

RLPRDO  Confidence intervals for the true response and for the average of a set of future observations on the response – out of core version.

RLRES  Perform a residual analysis for a fitted regression model.

LOCLIB

RLHPTS  Calculates the probability that a set of linear hypotheses about the coefficients of a regression model are simultaneously acceptable.

NAG

G02CEF  Takes selected elements from two vectors (typically vectors of means and standard deviations) to form two smaller vectors, and selected rows and columns from two matrices (typically either matrices of sums of squares and cross-products of deviations from means and Pearson product-moment correlation coefficients, or matrices of sums of squares and cross-products about zero and correlation-like coefficients) to form two smaller matrices, allowing re-ordering of elements in the process.

G02CFF  Re-orders the elements in two vectors (typically vectors of means and standard deviations), and the rows and columns in two matrices (typically either matrices of sums of squares and cross-products of deviations from means and Pearson product-moment correlation coefficients, or matrices of sums of squares and cross-products about zero and correlation-like coefficients).

NCARLB

CNFPRB  Calculates the probability that a linear combination of an alternate solution vector is a credible replacement for the same linear combination of calculated regression coefficients.
Categorical Data Analysis (Discrete Data Analysis) (see also LO1b)

**EDA**

CTBL
- Print a coded table of a matrix.

MEDPOL
- Analyze the two-way table in given data by median polish.

**IMSL**

BEMDP
- Median polish of a two-way table.

CBNRHO
- Estimation of the bivariate normal correlation coefficient using a contingency table.

CTLLF
- Log-linear fit of contingency table.

CTPR
- Compute exact probabilities for contingency tables.

CTRBYC
- Analysis of a contingency table.

GFIT
- Chi-squared goodness of fit test.

GTCN
- Sample size or number of class intervals determination for chi-squared test applications.

NHEXT
- Fisher's exact method for 2 by 2 tables.

**NAG**

G01AFF
- Performs the analysis of a two-way $r \times c$ contingency table or classification. If $r = c = 2$, and the total number of objects classified is 40 or fewer, then the probabilities for Fisher's exact test are computed. Otherwise, a test statistic is computed (with Yates' correction when $r = c = 2$), which under the assumption of no association between the classifications has approximately a chi-square distribution with $(r-1)(c-1)$ degrees of freedom.

**SAS (IBM 4341)**

CATMOD
- Procedure for categorical data modeling. It fits linear models to functions of response frequencies and can be used for linear modeling, log-linear modeling, logistic regression, and repeated measurement analysis. CATMOD uses maximum-likelihood estimation of parameters for log-linear models and the analysis of generalized logits or weighted-least-squares estimation of parameters for a wide range of general linear models. Options are documented in *SAS User's Guide: Statistics - Version 5*.

FREQ
- Produces one-way to n-way frequency and crosstabulation tables. For two-way tables, PROC FREQ computes tests and measures of association. For n-way tables, PROC FREQ does stratified analysis, computing statistics within as well as across strata. Frequencies can also be output to a SAS data set. Handles missing values and weighted observations. Documented in *SAS User's Guide: Basics - Version 5*. 
LOGIST
Fits the logistic multiple regression model to a single binary dependent variable or to an ordinal dependent variable. The procedure can fit one model or use either a backward elimination procedure or a stepwise technique. Documented in *SUGI Supplemental Library User's Guide - 1983 Edition* and "Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131".

MCSTRAT
Analyzes case-control studies where cases and controls are matched. PROC MCSTRAT estimates the odds ratio of a dichotomous disease state associated with a risk factor "adjusted" for confounding variables (if any). The procedure performs an iterative conditional maximum-likelihood fit of a logistic regression model. Documented in "Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131".

PROBIT

---

**L10a. Time Series: Transformations, Smoothing, Filtering**

**IMSL**

FTKALM
Kalman filtering.

FTRDIF
Transformations, differences and seasonal differences of a time series for model identification.

**LOCLIB**

FDPACD
Driver for LOCLIB packages FDPACN and FDPACR.

FDPACN
Package to generate the tap weights for a variety of finite-duration impulse response (also called FIR or non-recursive) discrete-time filters and plot the resulting filter characteristics on the line printer or microfilm plotter.

FDPACR
To generate a variety of continuous-time and discrete-time filter designs and plot the resulting filter characteristics on the line printer or the microfilm device.

**NAG**

G13AAF
Performs seasonal and non-seasonal differencing on a time series. Information which allows the original series to be reconstituted from the differenced series is also produced. This information is required in time series forecasting.

G13BAF
Filters a time series by an autoregressive moving average model.
**SAS (IBM 4341)**

**DIF**

**LAG**

**X11**
An adaptation of the Bureau of the Census X-11 Seasonal Adjustment program which can be used to seasonally adjust monthly or quarterly time series. Documented in *SAS/ETS User's Guide - Version 5.*

**SCILIB**

**FILTERG**
Computes a convolution of two vectors. This routine has been optimized for the Cray 1.

**FILTERS**
Computes a convolution of a symmetric filter vector with a data vector. This routine has been optimized for the Cray 1.

**OPFILT**
Solves the Wiener-Levinson system of equations $A x = b$, where $A$ is a Toeplitz matrix. The output is a vector of filter coefficients. This routine has been optimized for the Cray 1.

**STATLIB**

**MOVAVG**
Is a digital filtering routine for time series. It computes a $k$ term moving average, and also returns the moving average sampled every $k$th observation (to remove autocorrelation and spectral distortion caused by the moving average), and the original series minus the moving average (a high-pass filter operation). Also returned are values indicating the length of the returned series.

**TSDIFF**
Is a digital filtering routine for time series. It computes a stationary series by a differencing method.

---

**L10b. Time Series: Autocorrelation and Cross-Correlation, Univariate and Multivariate**

**IMSL**

**FTAUTO**
Mean, variance, autocovariances, autocorrelations, and partial autocorrelations for a stationary time series.

**FTCROS**
Means, variances, cross-covariances, and cross-correlations for two mutually stationary $n$ channel time series.

**FTCRXY**
Cross-covariance of two mutually stationary time series.

**FTFREQ**
Single or multichannel time series analysis in the time and frequency domains.
NAG

G13ABF Computes the sample autocorrelation function of a time series. It also computes
the sample mean, the sample variance and a statistic which may be used to test
the hypothesis that the true autocorrelation function is zero.

G13ACF Computes partial autocorrelation coefficients given a set of autocorrelation
coefficients. Also computes predictor error variance ratios for increasing orders of
autoregression, and autoregressive parameters associated with the predictor of
maximum order.

G13BCF Calculates cross correlations between two time series.

G13DAF Calculates the cross covariance or cross correlation function of a multivariate
time series.

G13DBF Calculates the multivariate partial autocorrelation function of a multivariate time
series.

STATLIB

ACORR Provides correlation analysis for time series. This routine computes autocorrela-
tion coefficients, and the standard error of each correlation and partial correlation
coefficient. Automatic printout includes lists and line-printer plots of both
coefficient series.

ACORRD Is a correlation analysis routine for time series, performing the same analysis as
STATLIB routine ACORR, but for a sequence of differenced series.

ACORRS Is a correlation analysis routine for time series, identical to STATLIB routine
ACORR, except that autocorrelation coefficients are returned to the user.

CCORRS Provides correlation analysis for time series. It computes cross-correlation
coefficients, and automatically produces a line-printer plot of the autocorrelation
function.

L10c. Time Series: ARMA and ARIMA Modeling and Forecasting

IMSL

FTARPS Preliminary estimation of the autoregressive parameters in an ARIMA stochastic
model.

FTCAST Time series forecasts and probability limits using an ARIMA (Box-Jenkins)
model.

FTCP Non-seasonal ARIMA (Box-Jenkins) stochastic model analysis for a single time
series with full parameter iteration and maximum likelihood estimation.

FTGEN Generation of a time series from a given ARIMA (Box-Jenkins) stochastic model.

FTMA Preliminary estimation of the moving average parameters in an arima stochastic
model.
FTML Maximum likelihood estimation of autoregressive and moving average parameters in an ARIMA (Box-Jenkins) stochastic model.

NAG

G13ADF Calculates preliminary estimates of the parameters of an autoregressive moving average (ARMA) model, from an autocorrelation function.

G13AEF Fits a seasonal autoregressive-integrated moving-average (ARIMA) model to an observed time series, using a non-linear least squares procedure incorporating backforecasting. Produces parameter estimates with standard errors, the residual series, and information for use in forecasting.

G13AFF Easy-to-use version of G13AEF.

G13AGF Accepts a series of new observations of a time series, the model of which is already fully specified (by G13AEF or G13AFE), and updates the “state set” information for use in constructing further forecasts.

G13AHF Produces forecasts of a time series, given a time series model already fitted by G13AEF or G13AFF.

G13AJF Derives the state set and a specified number of forecasts from a previously estimated seasonal ARIMA model.

SAS (IBM 4341)

ARIMA Analyzes and forecasts univariate time series data, transfer function data or intervention data using the autoregressive integrated moving average model developed by Box and Jenkins. Documented in SAS/ETS User’s Guide - Version 5.

STATLIB

ARIMAE Provides time series analysis with ARIMA models. This routine computes the least squares (maximum likelihood) estimates of the parameters in an ARIMA model. Automatic printout includes the correlation matrix; a plot and list of the residuals; the estimated parameter values and associated standard errors and confidence intervals; and details of each iteration until convergence.

ARIMAF Provides spectral analysis for time series. This routine computes minimum mean square error forecasts for a series using a fitted ARIMA model. Automatic printout includes a model summary, lists of the forecasts and confidence limits, a line-printer plot of the forecasts, and a list of weights used in the calculations.
L10d. Time Series: Transfer Function Modeling

**IMSL**

**FTTR**
Parameter estimates for a univariate transfer function model.

**NAG**

**G13BBF**
Filters a time series by a transfer function model.

**G13BDF**
Calculates preliminary estimates of the parameters of a transfer function model.

**G13BEF**
Estimates the parameters in a multi-input model relating one output series to one or more input series, using a choice of three slightly different estimation criteria: non-linear least squares, exact likelihood and marginal likelihood.

**G13BGF**
Accepts a series of new observations of an output time series and any associated input time series, for which a multi-input model is already specified, and updates the 'state-set' information for use in constructing further forecasts.

**G13BHF**
Produces forecasts of a time series (the output series) which depends on one or more other (input) series via a multi-input model which will usually have been fitted using G13BEF.

**G13BJF**
Produces forecasts of a time series (the output series) which depends on one or more other (input) series via a multi-input model for which the state set information is not available.

**SAS (IBM 4841)**

**ARIMA**
Analyzes and forecasts univariate time series data, transfer function data or intervention data using the autoregressive integrated moving average model developed by Box and Jenkins. Documented in *SAS/ETS User's Guide - Version 5*.

L10e. Time Series: Univariate Spectral Analysis

**IMSL**

**FTFREQ**
Single or multichannel time series analysis in the time and frequency domains.

**NAG**

**G13CAF**
Calculates the smoothed sample spectrum of a univariate time series using one of four lag windows - rectangular, Bartlett, Tukey, or Parzen window.

**G13CBF**
Calculates the smoothed sample spectrum of a univariate time series using spectral smoothing by the trapezium frequency (Daniell) window.
**NCARLIB**

SPAL Computes power spectral estimates for a one-dimensional stationary time series with options for detrending the time series data.

**SAS (IBM 4341)**

SPECTRA Produces estimates of the spectral and cross-spectral densities of a multivariate time series using a finite Fourier transform to obtain periodograms and cross periodograms. The periodogram ordinates can be smoothed by a moving average to produce estimated spectral and cross-spectral space densities. SPECTRA can also test whether or not the data are white noise. SPECTRA creates an output SAS data set whose variables contain values of the periodograms, cross periodograms, estimates of spectral densities, cross-spectral densities, amplitudes, squared coherency, and phase. Documented in *SAS/ETS User's Guide - Version 5*.

**STATLIB**

ACSPEC Provides spectral analysis for time series. This routine is the same as STATLIB routine ASPECS, except that the autocorrelation function (from STATLIB routine ACORRS, perhaps) is input along with the series.

ASPEC Provides spectral analysis for time series. This routine computes the autospectrum of the input series. The Fourier transform of the autocorrelation function is also computed, using a Tukey window and autocorrelation lags for smoothing. Automatic print-out is four line-printer, semi-log plots of the spectrum using different, automatically specified bandwidths. Each plot shows the 95% confidence interval for spectral estimates.

ASPECS Is a spectral analysis routine for time series, and is identical to STATLIB routine ASPEC except that the user specifies the number of lags (bandwidth) for each plot.

IASPEC Provides spectral analysis for time series. This routine produces a line-printer plot of the integrated sample periodogram showing critical lines for a 5% significance level test for departure from white noise.

**L10f. Time Series: Multivariate Spectral Analysis**

**IMSL**

FTFREQ Single or multichannel time series analysis in the time and frequency domains.
NAG

G13CCF Calculates the smoothed sample cross spectrum of a bivariate time series using one of four lag windows - rectangular, Bartlett, Tukey, or Parzen window.

G13CDF Calculates the smoothed sample cross spectrum of a bivariate time series using spectral smoothing by the trapezium frequency (Daniell) window.

G13CEF For a bivariate time series, calculates the cross amplitude spectrum, squared coherency, and lower and upper bounds for the univariate and bivariate (cross) spectra.

G13CFF For a bivariate time series, calculates the gain, phase, and lower and upper bounds for the univariate and bivariate (cross) spectra.

G13CGF For a bivariate time series, calculates the noise spectrum together with multiplying factors for the bounds and the impulse response function and its standard error from the univariate and bivariate (cross) spectra.

SAS (IBM 4341)

SPECTRA Produces estimates of the spectral and cross-spectral densities of a multivariate time series using a finite Fourier transform to obtain periodograms and cross periodograms. The periodogram ordinates can be smoothed by a moving average to produce estimated spectral and cross-spectral space densities. SPECTRA can also test whether or not the data are white noise. SPECTRA creates an output SAS data set whose variables contain values of the periodograms, cross periodograms, estimates of spectral densities, cross-spectral densities, amplitudes, squared coherency, and phase. Documented in SAS/ETS User's Guide - Version 5.

STATLIB

CCSPEC Is a spectral analysis routine for time series, and is identical to STATLIB routine CSPECS except that auto-correlations and cross-correlations must be input (perhaps from STATLIB routines ACORRS and CCORRS).

CSPEC Provides spectral analysis for time series. This routine produces four pairs of line-printer plots of the smooth phase and the arctanh of the squared coherency spectrums. Plots of the latter include 95% confidence intervals for a 5% significance level test of zero coherency.

CSPECS Is a spectral analysis routine for time series, identical to STATLIB routine CSPEC, except that the user must specify the number of plots and the number of lags (bandwidths) for each plot.
L101. Time Series: Other Time Series Techniques (see also L04a)

**IMSL**

**FTWEIN** Wiener forecast for a stationary stochastic process.

**FTWENM** Multichannel Wiener forecast.

**FTWENX** Maximum likelihood parameter estimates for a multichannel, single output time series model.

**NAG**

**G05EWF** Generates the next term from an autoregressive moving-average time series using a reference vector set up by G05EGF.

**SAS (IBM 4841)**

**AUTOREG** Estimates parameters in regression models when the data are time series and the error term is an autoregressive process. This procedure works with embedded missing values for the independent or dependent variables. Documented in *SAS/ETS User's Guide - Version 5*.

**FORECAST** Provides an automatic way to generate quickly forecasts of many time series. There are several other SAS procedures that can be used to produce better forecasts but at greater expense. Three methods are available: stepwise autoregressive, exponential smoothing, and winters method. Documented in *SAS/ETS User's Guide - Version 5*.

**MODEL** Used to compile a model to be processed later by PROC SYSLIN or PROC SIMLIN. Documented in *SAS/ETS User's Guide - Version 5*.

**PDLREG** Estimates regression parameters for time series data when the effects of the regressor variables can be distributed across time. PDLREG uses the Almon distributed lag polynomial with or without endpoint restrictions. The procedure can also produce estimates when a finite autoregressive error structure is specified. Documented in *SAS/ETS User's Guide - Version 5*.

**SIMLIN** Reads the coefficients for a set of linear structural difference equations, computes the reduced form, and uses the reduced form equations to generate predicted values. SIMLIN can only be applied to models that are: linear with respect to the parameters, linear with respect to the variables, square (as many equations as endogenous variables), nonsingular (the structural coefficients on the endogenous variables form an invertible matrix). Documented in *SAS/ETS User's Guide - Version 5*.

**STATESPACE** Can analyze and forecast stationary multivariate time series data or data that can be made stationary after differencing, including transfer function models that have random inputs. It does this by fitting a sequence of vector autoregressive models using the Yule-Walker equations and selects the order for which Akaike's information criterion is minimized. This order is then taken as the number of lags into the past to use in a canonical correlation analysis. Documented in *SAS/ETS User's Guide - Version 5*.
SYSLIN Estimates coefficients in an interdependent system of linear equations using any of the following methods: ordinary least squares, \(k\) class, minimum expected loss, two-stage least squares, limited-information maximum likelihood, three-stage least squares, seemingly unrelated regressions or joint generalized least squares. Documented in *SAS/ETS User's Guide - Version 5*.

SYSNLIN Combines iterative minimization methods for nonlinear regression with specialized estimation techniques for simultaneous equation systems to estimate parameters in a simultaneous system of nonlinear equations. Available estimation methods include: nonlinear ordinary least squares, nonlinear seemingly unrelated regression, nonlinear two-stage least squares, nonlinear three-stage least squares, iterated versions of the above that produce simultaneous estimates of the covariance matrix and the parameters. Documented in *SAS/ETS User's Guide - Version 5*.


L11a. Correlation Analysis: No Missing Values

IMSL


BECORI Estimates of means, standard deviations, and correlation coefficients (in-core version).

BECTR Tetrachoric correlation coefficient estimation.

BESRB Biserial and point-biserial correlation coefficients for a qualitatively dichotomized variable and a numerically measurable and classified variable.

BESRN Biserial correlation coefficient for a qualitatively dichotomized variable and a numerically or qualitatively classified variable.

CBNRHO Estimation of the bivariate normal correlation coefficient using a contingency table.

NMKN Kendalls test for correlation (rank correlation coefficient).

NMKSF Frequency distribution of \(k\) and the probability of equalling or exceeding \(k\), where \(k\), the total score from the Kendall rank correlation coefficient calculations, and \(n\), the sample size, are given.
NAG

G01ABF Computes the means, standard deviations, corrected sums of squares and products, maximum and minimum values, and the product-moment correlation coefficient for two variables. Unequal weighting may be given.

G02BAF Computes means and standard deviations of variables, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients for a set of data in the array $X$.

G02BDF Computes means and standard deviations of variables, sums of squares and cross-products about zero, and correlation-like coefficients for a set of data in the array $X$.

G02BGF Computes means and standard deviations of variables, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients for a set of data in specified columns of the array $X$.

G02BKF Computes means and standard deviations of variables, sums of squares and cross-products about zero, and correlation-like coefficients for a set of data in specified columns of the array $X$.

G02BNF Computes Kendall and/or Spearman non-parametric rank correlation coefficients for a set of data in the array $X$; the data array $X$ is overwritten, and on exit it contains the ranks of the observations.

G02BQF Computes Kendall and/or Spearman non-parametric rank correlation coefficients for a set of data in the array $X$; the data array $X$ is preserved, and the ranks of the observations are not available on exit from the routine.

SAS (IBM 4341)

CORR Computes correlation coefficients between variables, including Pearson product-moment and weighted product-moment correlations. Three nonparametric measures of association (Spearman's rank-order correlation, Kendall's tau-b and Hoeffding's measure of dependence, $D$) can also be produced as well as univariate descriptive statistics. Handles missing values and weighted observations. The correlations may be output to another SAS data set. Documented in SAS User's Guide: Basics - Version 5.

STATLIB

COREL Provides correlation analysis of a multivariate random sample. Automatic printout includes: simple and partial correlation coefficients and their significance levels, Spearman rank correlation coefficients, a test for quadratic relationship among the variables, and 95% and 99% confidence intervals for the simple correlation coefficients.

CORELS Provides correlation analysis of a multivariate random sample; it is identical to STATLIB routine COREL, except that it allows storage of the sample and partial correlation coefficient matrices.
L11b. Correlation Analysis: Missing Values

**IMSL**

**BECOVW** Means and variance-covariance or correlation matrix from data possibly containing missing observations, with weighting on option.

**BEMMI** Estimates of means, standard deviations, correlation coefficients, and third and fourth moments from a data matrix containing missing observations. (in-core version)

**BEMMO** Estimates of means, standard deviations, correlation coefficients, and third and fourth moments from a data matrix containing missing observations. (out of core version)

**NAG**

**G02BBF** Computes means and standard deviations of variables, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients for a set of data in the array X, omitting completely any cases with a missing observation for any variable.

**G02BCF** Computes means and standard deviations of variables, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients for a set of data in the array X, omitting cases with missing values from only those calculations involving the variables for which the values are missing.

**G02BEF** Computes means and standard deviations of variables, sums of squares and cross-products about zero, and correlation-like coefficients for a set of data in the array X, omitting completely any cases with a missing observation for any variable.

**G02BFF** Computes means and standard deviations of variables, sums of squares and cross-products about zero and correlation-like coefficients for a set of data in the array X, omitting cases with missing values from only those calculations involving the variables for which the values are missing.

**G02BHF** Computes means and standard deviations of variables, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients for a set of data in specified columns of the array X, omitting completely any cases with a missing observation for any variable (either over all variables in the data array or over only those variables in the selected subset).

**G02BJF** Computes means and standard deviations of variables, sums of squares and cross-products of deviations from means, and Pearson product-moment correlation coefficients for a set of data in specified columns of the array X, omitting cases with missing values from only those calculations involving the variables for which the values are missing.

**G02BLF** Computes means and standard deviations of variables, sums of squares and cross-products about zero, and correlation-like coefficients for a set of data in specified columns of the array X, omitting completely any cases with a missing observation for any variable (either over all variables in the data array or over only those variables in the selected subset).
G02BMF Computes means and standard deviations of variables, sums of squares and cross-products about zero, and correlation-like coefficients for a set of data in specified columns of the array $X$, omitting cases with missing values from only those calculations involving the variables for which the values are missing.

G02BPF Computes Kendall and/or Spearman non-parametric rank correlation coefficients for a set of data in the array $X$, omitting completely any cases with a missing observation for any variable; the data array $X$ is overwritten, and on exit contains the ranks of the observations.

G02BRF Computes Kendall and/or Spearman non-parametric rank correlation coefficients for a set of data in the array $X$, omitting completely any cases with a missing observation for any variable; the data array $X$ is preserved, and the ranks of the observations are not available on exit from the routine.

G02BSF Computes Kendall and/or Spearman non-parametric rank correlation coefficients for a set of data in the array $X$, omitting cases with missing values from only those calculations involving the variables for which the values are missing; the data array $X$ is preserved, and the ranks of the observations are not available on exit from the routine.

SAS (IBM 4841)

CORR Computes correlation coefficients between variables, including Pearson product-moment and weighted product-moment correlations. Three nonparametric measures of association (Spearman's rank-order correlation, Kendall's tau-b and Hoeffding's measure of dependence, $D$) can also be produced as well as univariate descriptive statistics. Handles missing values and weighted observations. The correlations may be output to another SAS data set. Documented in SAS User's Guide: Basics - Version 5.

L12. Discriminant Analysis

IMSL

ODFISH Linear discriminant analysis method of Fisher for reducing the number of variables.

ODNORM Multivariate normal linear discriminant analysis among several known groups.

SAS (IBM 4841)

CANDISC Performs a canonical discriminant analysis, computes Mahalanobis distances, and does both univariate and multivariate one-way analyses of variance. Output data sets containing canonical coefficients and scores on the canonical variables can be created. Observations can be weighted. Documented in SAS User's Guide: Statistics - Version 5.

DISCRIM Computes linear or quadratic discriminant functions for classifying observations into two or more groups on the basis of one or more numeric variables. The
NEIGHBOR Performs a nearest neighbor discriminant analysis, classifying observations into
groups according to either the nearest neighbor rule or the \( k \)-nearest-neighbor rule. Documented in *SAS User’s Guide: Statistics - Version 5*.

STEPDISC Performs a stepwise discriminant analysis by forward selection, backward elimi-
nation, or stepwise selection of variables that can be useful for discriminating

---

**L13a. Factor Analysis**

**IMSL**

**OFCOEF** Compute a matrix of factor score coefficients for input to IMSL routine OFS-
COR.

**OFCOMM** Compute an unrotated factor loading matrix according to a common factor model
by unweighted or generalized least squares, or by maximum likelihood pro-
cedures.

**OFHARR** Transformation of unrotated factor loading matrix to oblique axes by Harris-
Kaiser method.

**OFIMAG** Compute an unrotated factor loading matrix according to an image model.

**OFPRI** Compute an unrotated factor loading matrix according to a principal component
model.

**OFPROT** Oblique transformation of the factor loading matrix using a target matrix, includ-
ing pivot and power vector options.

**OFRESI** Communalities and normalized factor residual correlation matrix calculation.

**OFROTA** Orthogonal rotation of a factor loading matrix.

**OFSCHN** Orthogonal transformation of the factor loading matrix using a target matrix.

**OFSCOR** Compute a set of factor scores given the factor score coefficient matrix.

**SAS (IBM 4341)**

**FACTOR** Performs several types of common factor and component analysis. Both orthogo-
nal and oblique rotations are available. Scoring coefficients can be computed by
the regression method, and estimated factor scores can be written to an output
data set. All major statistics computed by the procedure can also be saved in an
output data set. Options are documented in *SAS User’s Guide: Statistics - Ver-
sion 5*. 
L13b. Principal Component Analysis

IMSL

OPRINC Principal components of a multivariate sample of observations.

SAS (IBM 4341)

PRINCOMP Performs principal component analysis for either the correlation or covariance matrix. Output data sets containing eigenvalues, eigenvectors, and standardized or unstandardized principal components scores can be created. Documented in SAS User’s Guide: Statistics - Version 5.

L13c. Canonical Correlation Analysis

SAS (IBM 4341)


L14. Cluster Analysis

IMSL

OCDIS Pairwise Euclidean distances between the columns of a matrix.

OCLINK Perform a single-linkage or complete-linkage hierarchical cluster analysis given a similarity matrix.

USTREE Print a binary tree (which may represent the output of a clustering algorithm in chapter 0).
SAS (IBM 4341)

ACECLUS  Obtains approximate estimates of the pooled within-cluster covariance matrix when the clusters can be assumed multivariate normal with equal covariance matrices. Neither cluster membership nor the number of clusters need be known. ACECLUS can produce output data sets containing the approximate within-cluster covariance estimate, eigenvalues and eigenvectors from a canonical analysis, and canonical variable scores. Documented in SAS User's Guide: Statistics - Version 5.

CLUSTER  Hierarchically clusters the observations in a SAS data set using one of eleven methods. The data can be numeric coordinates or distances. CLUSTER creates an output data set from which the TREE procedure can draw a tree diagram or output clusters at a specified level of the tree. Observations can be weighted by an integer with the FREQ statement. Documented in SAS User's Guide: Statistics - Version 5.

FASTCLUS  Designed for disjoint clustering of very large data sets and can find good clusters with only two or three passes over the data. You specify the maximum number of clusters and optionally, the minimum radius of the clusters. The procedures can produce an output data set containing a cluster membership variable as well as an output data set containing cluster means. Options are documented in SAS User's Guide: Statistics - Version 5.


TREE  Prints a tree diagram, also known as a dendrogram or phenogram, using a data set created by the CLUSTER or VARCLUS procedure. PROC TREE can also create an output data set identifying disjoint clusters at a specified level in the tree. Documented in SAS User's Guide: Statistics - Version 5.

VARCLUS  Performs either disjoint or hierarchical clustering of variables based on a correlation or covariance matrix. Clusters are chosen to maximize the variation accounted for by either the first principal component or the centroid component of each cluster. An output data set containing the results of the analysis can be created and used with the SCORE procedure to compute cluster component scores. Documented in SAS User's Guide: Statistics - Version 5.
L15. Life Testing, Survival Analysis

IMSL

CLIFE Life table analysis.

SAS (IBM 4341)


LIFETEST Used with data that may be right censored to compute nonparametric estimates of the survival distribution and to compute rank tests for association of the response variable with other variables. Documented in SAS User's Guide: Statistics - Version 5.


SURVDIFF Provides nonparametric statistics to compare survival curves in independent sample. PROC SURVDIFF replaces the SURVTEST procedure and complements SURVFIT, which estimates survival curves. Documented in "Changes and Enhancements in the Version 5 SUGI Supplemental Library - Technical Report: S-131".


L16. Exploratory Data Analysis

EDA

CINIT Initialization routine, to be called at start of any main program which calls one of the EDA subroutines (either directly or indirectly).

STMNLF Produces a stem-and-leaf display of data on a character printing device.

LVALS For a batch of values, finds selected quantiles known as letter values.

BOXES On the line printer, print adjacent boxplots on a single scale for a given set of values.

PLOT Plot ordered pairs in a condensed manner on the line printer.
NCAR Software Catalog

Performs nonlinear smoothing.
For given data, fit a straight line by the "resistant line" technique.

Print a coded table of a matrix.

Analyze the two-way table in given data by median polish.
Perform computations for a suspended rootogram.
Print, bin by bin, the observed count, the raw residual, the double-root residual, and an abbreviated display of the double-root residual.
Given a batch of values in increasing order, and a set of bin boundaries, determine the bin counts.
Print a horizontal axis for a boxplot or regular plot on the line printer.

IMSL

Median polish of a two-way table.
Print a boxplot (k samples).
Print a stem and leaf display.

SAS (IBM 4341)


Produces simple descriptive statistics for numeric variables which provide more detail on the distribution of a variable. These statistics include extreme values of a variable, quantiles, stem-and leaf plot, box plot, normal probability plot, frequency table, a test for normality, the number of observations, number of missing observations, mean, sum, standard deviation, variance, skewness, kurtosis, sum of the weights, largest value, smallest value, range, mode, and signed rank statistic. Observations may be weighted and missing values can be excluded. Documented in SAS User's Guide: Basics - Version 5.

L17. Sampling Techniques

IMSL

Generate a simple random sample from a finite population.
Simple random sampling with proportion data-inferences regarding the population proportion and total.
Stratified random sampling with proportion data – inferences regarding the population proportion and total.
SSRAND  Simple random sampling with continuous data – inferences regarding the population mean and total using ratio or regression estimation.

SSRBLK  Stratified random sampling with continuous data – inferences regarding the population mean and total using ratio or regression estimation.

SSSAND  Simple random sampling with continuous data – inferences regarding the population mean and total.

SSSBLK  Stratified random sampling with continuous data – inferences regarding the population mean and total.

SSSCAN  Single stage cluster sampling with continuous data – inferences regarding the population mean and total.

SSSEST  Two-stage sampling with continuous data and equisized primary units – inferences regarding the population mean and total.

L20. Other Statistical Techniques

SAS (IBM 4341)


INBREED  Calculates the covariance or inbreeding coefficients for a pedigree. INBREED is unique in that it handles very large populations and has a special mode of operation for analysis within nonoverlapping generations. Documented in SUGI Supplemental Library User's Guide - 1983 Edition.


PROX  Calculates distances between variables, with observation values representing different dimensions over which the distances are being calculated. PROC PROX produces a SAS data set containing distance values that can be used in other SAS procedures, such as ALSCAL. Documented in SUGI Supplemental Library User's Guide - 1983 Edition.
SCORE Produces scores by multiplying values from two SAS data sets, one containing coefficients (for example, factor-scoring coefficients or regression coefficients) and the other containing the original data used to calculate the coefficients. The result of this multiplication is an output SAS data set containing the linear combinations of the coefficients and the original data values. Documented in *SAS User's Guide: Statistics - Version 5.*

### M1a1. Sorting of Integer Data

#### NAG

**M01ACF** Sorts an array of pointers via a detached key sort to give the ascending order of a given integer array.

**M01ADF** Sorts an array of pointers via a detached key sort to give the descending order of a given integer array.

**M01AGF** Sorts the rows of an integer matrix into ascending order of an index column.

**M01AHF** Sorts the rows of an integer matrix into descending order of an index column.

**M01ALF** Sorts a vector of integer numbers into ascending order and provides an index indicating the position of the sorted numbers in the original array.

**M01AMF** Sorts a vector of integer numbers into descending order and provides an index indicating the position of the sorted numbers in the original array.

**M01AQF** Sorts a vector of integer numbers into ascending order.

#### NCARLB

**SSORT** A fast, stable in-core sorting routine that allows the reordering of the rows of an array based on the alphanumeric (collating sequence) or numeric (integer or real) values in any column of the array.

#### PORT

**SRTAI** Sorts in ascending order an array of integer data. Shell's method is used.

**SRTDI** Sorts in descending order an array of integer data. Shell's method is used.

**SRTPAI** Performs a sort in ascending order of a vector of integer data. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell's method is used.

**SRTPDI** Performs a sort in descending order of a vector of integer data. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell's method is used.

**SRTRI** Uses a given permutation of the integers 1, \ldots, N stored in a permutation vector to rearrange integer data in another input vector.

File Management 282
SAS (IBM 4341)

SORT
Sorts observations in a SAS data set by one or more variables, storing the resulting sorted observations in a new SAS data set or replacing the original. Options documented in *SAS User's Guide: Basics - Version 5*.

SLATEC

ISORT
Sorts integer array \( X \) and optionally makes the same interchanges in integer array \( Y \). The array \( X \) may be sorted in increasing order or decreasing order. A slightly modified quicksort algorithm is used.

UTILITIES

SORT
VOS Software Tools command for sorting and/or merging.

M1a2. Sorting of Real Data

IMSL

VSAR
Sorting of matrices (with options)

VSODA
Sorting of columns of a double precision matrix in ascending order of keys in rows.

VSORA
Sorting of columns of a real matrix into ascending order of keys in rows.

VSRTA
Sorting of arrays by algebraic value.

VSRTM
Sorting of arrays by absolute value.

VSRTP
Sorting of arrays by absolute value – permutations returned.

VSRTR
Sorting of arrays by algebraic value – permutations returned.

VSRTP
Interchange the rows or columns of a matrix using a permutation vector such as the one obtained from IMSL routines VSRTP or VSRTR.

NAG

E02ZAF
Sorts two-dimensional data into rectangular panels.

M01AAF
Sorts an array of pointers via a detached key sort to give the ascending order of a given real array.

M01ABF
Sorts an array of pointers via a detached key sort to give the descending order of a given real array.

M01AEF
Sorts the rows of a real matrix into ascending order of an index column.

M01AFF
Sorts the rows of a real matrix into descending order of an index column.
M01AJF Sorts a vector of real numbers into ascending order and provides an index indicating the position of the sorted numbers in the original array.

M01AKF Sorts a vector of real numbers into descending order and provides an index indicating the position of the sorted numbers in the original array.

M01ANF Sorts a vector of real numbers into ascending order.

M01APF Sorts a vector of real numbers into descending order.

M01ARF Sorts a vector of integer numbers into descending order.

NCARLB

SSORT A fast, stable in-core sorting routine that allows the reordering of the rows of an array based on the alphanumeric (collating sequence) or numeric (integer or real) values in any column of the array.

PORT

SRTAD Sorts in ascending order an array of double precision data. Shell's method is used.

SRTAR Sorts in ascending order an array of real data. Shell's method is used.

SRTDD Sorts in descending order an array of double precision data. Shell's method is used.

SRTDR Sorts in descending order an array of real data. Shell's method is used.

SRTPAD Performs a sort in ascending order of a vector of double precision data. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell's method is used.

SRTPAR Performs a sort in ascending order of a vector of real data. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell's method is used.

SRTPDD Performs a sort in descending order of a vector of double precision data. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell's method is used.

SRTPDR Performs a sort in descending order of a vector of real data. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell's method is used.

SRTRD Uses a given permutation of the integers 1, . . ., N stored in a permutation vector to rearrange double precision data in another input vector.

SRTRR Uses a given permutation of the integers 1, . . ., N stored in a permutation vector to rearrange real data in another input vector.
**SAS (IBM 4841)**

**SORT**
Sorts observations in a SAS data set by one or more variables, storing the resulting sorted observations in a new SAS data set or replacing the original. Options documented in *SAS User's Guide: Basics - Version 5.*

**SLATEC**

**SSORT**
Sorts array X and optionally makes the same interchanges in array Y. The array X may be sorted in increasing order or decreasing order. A slightly modified quicksort algorithm is used.

---

**M1a3. Sorting of Character Data**

**NAG**

**M01BAF**
Sorts a vector containing character data into reverse alphabetic order.

**M01BBF**
Sorts a vector containing character data into alphabetic order.

**M01BCF**
Sorts selected columns (records) of an integer array containing character data, so that the elements of index rows (keys) are in reverse alphabetic order.

**M01BDF**
Sorts selected columns (records) of an integer array containing character data, so that the elements of index rows (keys) are in alphabetic order.

**NCARLB**

**SSORT**
A fast, stable in-core sorting routine that allows the reordering of the rows of an array based on the alphanumeric (collating sequence) or numeric (integer or real) values in any column of the array.

**PORT**

**SRTAH**
Sorts in ascending order an array of Hollerith data in accordance with the collating sequence for the computer being used. Shell’s method is used.

**SRTDH**
Sorts in descending order an array of Hollerith data in accordance with the collating sequence for the computer being used. Shell’s method is used.

**SRTPAH**
Performs a sort in ascending order of an array of Hollerith data, in accordance with the collating sequence for the computer being used. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell’s method is used.

**SRTPDH**
Performs a sort in descending order of an array of Hollerith data, in accordance with the collating sequence for the computer being used. The data themselves are not moved, but a permutation required to bring the data into sorted order is computed and stored in a permutation vector. Shell’s method is used.
SRTRH  Uses a given permutation of the integers 1, \ldots, N stored in a permutation vector to rearrange Hollerith data in another input vector.

\textit{SAS (IBM 4341)}

SORT  Sorts observations in a SAS data set by one or more variables, storing the resulting sorted observations in a new SAS data set or replacing the original. Options documented in \textit{SAS User's Guide: Basics - Version 5}.

\textbf{UTILITIES}

SORT  VOS Software Tools command for sorting and/or merging.
TSORT  VOS Software Tools command for topologically sorting symbols.

\textbf{M2. Conversion, Packing and Unpacking}

\textbf{AMOSLIB}

DATAD  Converts a single dimension array in \textit{D}-format into a \textit{FORTRAN} data statement in \textit{E}-format carrying 15 significant digits.

\textbf{LOCLIB}

CCONV  Supports LOCLIB package \textit{FCONV} with non-user entries.
CHCONVF  A package for converting ASCII characters to DPC characters and DPC characters to ASCII characters.
FCONV  Performs 7600-to-Cray mixed-mode record conversion.
HEXER  A main program that reads standard graphics metacode (as described in the NCAR Graphics Manual) and writes a formatted hexadecimal dump of the metacode, with a special carriage control character duplicated at the beginning of each line. Can be used to send graphics files to an RJE site.
RPTIN  Reads logical records from a tape made by \textit{RPTOUT}, including those made on CDC 7600.
RPTOUT  Packs short variable-length logical records into larger physical records on tape.
UBLOK  Unblocks fixed length logical records from larger physical records. Logical records must not overlap word boundaries.
UNHEXER  A main program that reads a file containing lines of text (produced by \textit{HEXER}, for example), extracts any embedded hex-encoded metacode file, and outputs the equivalent standard binary metacode file to unit 8.
UZBLOK  Unblocks fixed length logical records from larger physical records. Logical records may overlap word boundaries.
NCARLB

FPACKER A package of real number to integer packing routines which are able to pack numbers in the ratio of 1 to 2, 3, or 4.

RUGRID Unpacks and reads National Meteorological Center grid data.

PORT

CNVBDC Converts a double precision vector to a complex vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBDI Converts a double precision vector to an integer vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBDR Converts a double precision vector to a real vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBIC Converts an integer vector to a complex vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBID Converts an integer vector to a double precision vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBIR Converts an integer vector to a real vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBRC Converts a real vector to a complex vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBRD Converts a real vector to a double precision vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVBRI Converts a real vector to an integer vector using a backward loop (which has a different effect than a forward loop if the vectors overlap).

CNVFDC Converts a double precision vector to a complex vector using a forward loop.

CNFDDI Converts a double precision vector to an integer vector using a forward loop.

CNVFDI Converts a double precision vector to a real vector using a forward loop.

CNVFIC Converts an integer vector to a complex vector using a forward loop.

CNVFID Converts an integer vector to a double precision vector using a forward loop.

CNVFIR Converts an integer vector to a real vector using a forward loop.

CNVFRC Converts a real vector to a complex vector using a forward loop.

CNVFRI Converts a real vector to a double precision vector using a forward loop.

CNVFRI Converts a real vector to an integer vector using a forward loop.

SLATEC

D9PAK Packs a base 2 exponent into a double precision floating point number.

D9UPAK Unpacks a double precision floating point number $z$ so that $z = y \cdot 2^n$.

R9PAK Packs a base 2 exponent into a floating point number.

R9UPAK Unpacks a floating point number $z$ so that $z = y \cdot 2^n$. 
**S88FMT** Converts integer to character.

**PACK** Packs a specified number of words into a packed list.

**UNPACK** Expands full words of data into a larger number of partial words.

**UTILITIES**

**CPRESS** VOS Software Tools command for compressing runs of repeated characters in input files. The output file can later be expanded with the tool EXPAND.

**DELTA** VOS Software Tools command for making a delta with respect to tools history files.

**ENTAB** VOS Software Tools command for converting spaces to tabs and spaces.

**EOFILT** Locally developed Cray command for converting EOF markers in CRAY permanent datasets to another string, and reconverting the string occurrences back to EOFs. EOFILT may also be used to rid a permanent dataset of EOFs.

**EXPAND** VOS Software Tools command for expanding files previously compressed by tool 'CPRESS'.

**HEXER** Locally developed Cray command which reads standard graphics metacode (as described in the NCAR Graphics Manual) and writes a formatted hexadecimal dump of the metacode, with a special carriage control character duplicated at the beginning of each line. Can be used to send graphics files to an RJE site.

**OS** VOS Software Tools command for overstriking, i.e. converting backspaces into multiple lines.

**PLIBCONV** Locally developed Cray command which converts 7600 PLIB volumes to UPDATE volumes.

**TBMCONV** Locally developed Cray command program that reads TBM volumes written by the CDC 7600, including volumes disposed from the Cray to the TBM via the 7600. TBMCONV unpacks the original records, converts them to Cray records, and writes the resulting records to a user-specified Cray dataset.

---

**M4. Character Manipulation, Bit Operations**

**LOCLIB**

**BYTES** Cray CAL version of old 7600 system library program for unpacking bytes.

**CHCONVF** A package for converting ASCII characters to DPC characters and DPC characters to ASCII characters.

**FNDCHR** Searches through an input character string to find a given character string.
NCARLB

ENCD
ENCD is used by VELVEC and the CONREC routines to generate ASCII labels from numeric values by encoding them in an appropriate format.

GBYTES
Cray FORTRAN version of old 7600 system library program for unpacking bytes.

SUPPORT12C
SUPPORT12C is a collection of CAL routines which support plotting and may be of general use for character and bit manipulation. Its entry points are GETCHR, IAND, INTT, IOR, ISHIFT, LOC, PACKUM, SETCHR. These are described in the NCAR Graphics Software Manual.

SUPPORT12F
SUPPORT12F is a collection of FORTRAN routines which support plotting and may be of general use for character and bit manipulation. Its entry points are ENCODE, PERROR, ULIBER, and WRITEB. These are described in the NCAR Graphics Software Manual.

SYSLIB

CEXPR
Transforms an expression character string (one right-justified character per word) to a reverse polish table.

CHKEDRT
Checks the range of the ‘ED’ and ‘RT’ parameters used typically in calls to PDM functions. Returns the binary conversion if in range.

CRACK
Reformats (cracks) a user-supplied string into verb, separators, keywords, and values. The cracked directive is placed in a user-supplied buffer.

DTTS
Converts from date and time to timestamp.

EEXPR
Evaluates an expression which has been cracked by the companion routine CEXPR.

MTTS
Converts from a real time value to the corresponding timestamp value.

NUMBER
Performs numeric input conversion specifically, assemble integer number from R1 character string.

RBN
Converts trailing blanks to nulls.

RCW
Contains entries for reading characters and words, in partial and full record mode.

RNB
Converts trailing nulls to blanks. Includes entries $RNB (callable from CAL only), SFN, and RNB.

TSDT
Converts between timestamps and the date and time as ASCII strings.

TSMT
Converts from a timestamp to the corresponding real-time clock value.

UTILITIES

FIELD
VOS Software Tools command for manipulating fields of data.

KWIC
VOS Software Tools command for making keyword-in-context index.

LL
VOS Software Tools command for printing the lengths of the shortest and longest lines in the input files.
TR
VOS Software Tools command for character transliteration.

UNROT
VOS Software Tools command for processing the rotated output of tool ‘KWIK’ to generate a keyword-in-context index.

M5. Searching

IMSL
USMNMX Determination of the minimum and maximum values of a vector.

LOCBL
FNDCHR Searches through an input character string to find a given character string.

NCARLB
BSERCH Performs binary search of a table of floating point numbers.

PORT
INTRVD Finds the interval in a sorted double precision array to which an element belongs.
INTRVI Finds the interval in a sorted integer array to which an element belongs.
INTRVR Finds the interval in a sorted real array to which an element belongs.
MONOD Tests if a double precision vector is monotone increasing or decreasing.
MONOI Tests if an integer vector is monotone increasing or decreasing.
MONOR Tests if a real vector is monotone increasing or decreasing.
SMONOD Tests if a double precision vector is strictly monotone increasing or decreasing.
SMONOI Tests if an integer vector is strictly monotone increasing or decreasing.
SMONOR Tests if a real vector is strictly monotone increasing or decreasing.

SCILIB
ISRCHEQ Returns the first location in a real (or integer) array that is equal to the real (or integer) target.
ISRCCHFGE Returns the first location in a real array that is greater than or equal to the real target.
ISRCCHFGT Returns the first location in a real array that is greater than the real target.
ISRCCHFLE Returns the first location in a real array that is less than or equal to the real target.
ISRCHFLT  Returns the first location in a real array that is less than the real target.
ISRCHIGE  Returns the first location in an integer array that is greater than or equal to an integer target.
ISRCHIGT  Returns the first location in an integer array that is greater than an integer target.
ISRCHILE  Returns the first location in an integer array that is less than or equal to an integer target.
ISRCHILT  Returns the first location in an integer array that is less than an integer target.
ISRCHNE  Returns the first location in a real (or integer) array that is not equal to the real (or integer) target.
ORDERS  Sorts fixed-length records. It uses the radix sort, more commonly known as a bucket or pocket search. It is optimized for the Cray computer.
OSRCHF  Returns the index of the first location in an ordered real array that contains the target (type real).
OSRCHI  Returns the index of the first location in an ordered integer array that contains the target (type integer).
WHENEQ  Returns all locations in a real (or integer) array that are equal to the real (or integer) target.
WHENFGE  Returns all locations in a real array that are greater than or equal to the real target.
WHENFGT  Returns all locations in a real array that are greater than the real target.
WHENFLE  Returns all locations in a real array that are less than or equal to the real target.
WHENFLT  Returns all locations in a real array that are less than the real target.
WHENIGE  Returns all locations in an integer array that are greater than or equal to an integer target.
WHENIGT  Returns all locations in an integer array that are greater than an integer target.
WHENILE  Returns all locations in an integer array that are less than or equal to an integer target.
WHENILT  Returns all locations in an integer array that are less than an integer target.
WHENNE  Returns all locations in a real (or integer) array that are not equal to the real (or integer) target.

UTILITIES

FB  VOS Software Tools command for searching blocks of lines for text patterns. It is similar to tool ‘FIND’ except that if a pattern is found, the entire block is copied to standard output, rather than simply the single line in which the pattern occurred.
FIND  VOS Software Tools command for searching a file for text patterns.
N. Debugging

**LOCLIB**

**CONCORD**  Source for Cray binary module CNCRD.

**PFORTCF**  Fortran source for PFORT on the Cray, a Fortran '66 standards checker that also checks for some kinds of errors that compilers can't catch.

**PORT**

**FDUMP**  Dump routine called by PORT routine SETERR.

**SYSLIB**

**DUMPJOB**  Creates an unblocked dataset containing the user job area image (including register states). This data is suitable for input to the DUMP or DEBUG programs.

**SLERP**  Cray-1 routine for processing errors in $SYSLIB library routines. Aborts with traceback. Call from CAL only.

**SNAP**  Copies current register contents to $OUT.

**SNAPDMP**  Copies current register contents to $OUT. The A and S registers are always dumped. One can also copy desired registers $B$, $T$, or $V$ in a specified format.

**TRBK**  Prints a list of all subroutines active in the current calling tree. The user may specify a unit to receive the list. If no unit is specified, the list will be printed to the user logfile.

**TRBKLV**  Aids the trace back mechanism by returning information for the current level.

**UTILITIES**

**CONCORD**  Externally acquired Cray command which provides a concordance for one or more programs and subprograms.

**DEBUG**  Cray system control statement for interpreting a dump.

**DSDUMP**  Cray system control statement for dumping specified portions of a dataset to another in blocked or unblocked format.

**DUMP**  Cray system control statement for reading and formatting selected parts of the memory image, and writes the information onto another dataset.

**DUMPJOB**  Cray system control statement for copying current memory image to $DUMP$.

**FLDUMP**  Cray system control statement for recovering and dumping flowtrace tables when a program aborts with flow tracing active.

**PDSDUMP**  Cray system control statement for dumping specified permanent datasets to a dataset.

**SID**  Cray system control statement for debugging in interactive or batch mode.

**TEST**  Locally developed Cray command to test some of NCAR's public software.
Q1. Date and Time, Program Timers

**NCARLB**

- **DATEJ** Computes date information from number of hours after December 31, 1920.
- **HOURS** Computes hours since December 31, 1920, from date information.

**SYSLIB**

- **DTTS** Converts from date and time to timestamp.
- **MTTS** Converts from a real time value to the corresponding timestamp value.
- **TREMAIN** Cray-1 routine for returning seconds remaining for job execution.
- **TSDT** Converts between timestamps and the date and time as ASCII strings.
- **TSMT** Converts from a timestamp to the corresponding real-time clock value.
- **UNITTS** Returns the number of timestamp units in a specified number of standard time units.

**UTILITIES**

- **DATE** VOS Software Tools command for printing the date.

Q3. Machine and System Dependent Constants

**IMSL**

- **UGETIO** To retrieve current values and to set new values for input and output unit identifiers.

**NAG**

- **X01AAF** Returns the value of the constant $\pi$.
- **X01ABF** Returns the value of Euler’s constant, $\lambda$.
- **X02AAF** Returns the value $\varepsilon$, where $\varepsilon$ is the smallest positive real number such that $1.0 + \varepsilon > 1.0$.
- **X02ABF** Returns the value of $\text{RMIN}$, the smallest positive real floating-point number such that $\text{RMIN}$ and $-\text{RMIN}$ are exactly representable on the computer.
- **X02ACF** Returns the value of $\text{RMAX}$ where $\text{RMAX}$ is the largest computable real floating-point number such that $\text{RMAX}$ and $-\text{RMAX}$ are exactly representable on the computer.
NCAR Software Catalog

X02ADF  Returns the value of TOL, the ratio RMIN/EPS (see X02ABF and X02AAF).

X02AEF  Returns the largest negative real value ENEG such that exp(ENEG) can be successfully evaluated without underflow by the compiler-supplied EXP routine (DEXP for double precision implementations).

X02AFF  Returns largest positive permitted argument for EXP (or DEXP).

X02AGF  Returns the value of the smallest positive representable floating-point number R such that -R, 1.0/R and -1.0/R are all computable without underflow or overflow.

X02AHF  Returns largest positive permitted argument for SIN and COS (or DSIN and DCOS).

X02BAF  Returns the value of BASE, the base of the arithmetic used on the computer.

X02BBF  Returns the value of MAXINT, the largest integer such that MAXINT and -MAXINT are representable on the computer.

X02BCF  Returns the value of MAXPW2, the largest integer power to which 2.0 may be raised without overflow.

X02BDF  Returns the value of MINPW2, the largest negative integer power to which 2.0 may be raised without underflow.

X02BEF  Returns the value of MAXDEC, the maximum number of decimal digits which can be accurately represented in the computer over the whole range of floating-point numbers.

X02CAF  In a non-paged environment returns zero. In a paged environment, it returns a safe underestimate of the amount of actual storage (as opposed to virtual storage) that is expected to be available to a program running under typical conditions (i.e. the expected "active set size"); the estimate is given as a number of real variables; the value may vary between installations.

X02DAF  Returns .FALSE. if the system sets underflowing quantities to zero, without any error indication or undesirable warning or system overhead.

X04AAF  Returns the value of the current error message unit number, or sets the current error message unit number to a new value.

X04ABF  Returns the value of the current advisory message unit number, or sets the current advisory message unit number to a new value.

NCARLB

MACHCR  Includes the functions I1MACH, R1MACH, and D1MACH that return various environment parameters which may be used in tailoring a program to the Cray without making it unportable. Parameters that describe the floating point number system, the allowable range for integers, word and character sizes, and standard input and output units are included.

PORT

D1MACH  Provides the machine-dependent double-precision constants required to adapt PORT programs to individual computers.

I1MACH  Provides the machine-dependent integer constants required to adapt PORT programs to individual computers.
NCAR Software Catalog

Category Q3

R1MACH Provides the machine-dependent real constants required to adapt PORT programs to individual computers.

SCILIB

SMACH Real function of an integer argument which returns Cray 1 machine constants (arguments 1,2,3 yield machine epsilon, smallest normalized representable number, largest normalized representable number, respectively).

SLATEC

D1MACH Returns double precision machine dependent constants.
I1MACH Returns integer machine dependent constants.
R1MACH Returns single precision machine dependent constants.
XERFRT Prints error messages.
XERSAV Records that an error occurred.

Q4. Documentation Retrieval

IMSL

UHELP Display methods of obtaining information on IMSL conventions regarding various subjects and provide a means for individual sites to supply users with site specific information.
UHELP1 Write information regarding IMSL conventions and notation to an output file.
UHELP2 Write information regarding IMSL input and output conventions.
UHELP3 Write information regarding IMSL error detecting facilities.
UHELP4 Write information regarding matrix/vector storage modes used in IMSL subroutines.

UTILITIES

GETBIN Locally developed Cray command for accessing old, current, or new versions of public binary libraries specific to the Cray.
GETDOC Locally developed Cray command for documentation retrieval of NCAR’s public libraries specific to the Cray.
GETSRC Locally developed Cray command for source retrieval of NCAR’s public libraries.
Q5. Error Handling Packages

**AMOSLIB**

**ERRCHK** Processes diagnostics and warning messages which originate in the AMOSLIB mathematical program library routines.

**FUNPACK**

**FCNMON** Concentrates all FUNPACK error diagnostic facilities and print statements; intended for use only by other FUNPACK elements.

**FPGTER** Diagnoses FUNPACK computed GOTO out-of-range errors.

**MONERR** Monitors FUNPACK errors.

**IMSL**

**UERSET** Set message level for IMSL routine UERTST.

**UERTST** Print a message reflecting an error condition.

**NAG**

**P01AAF** Returns the value of IERROR or terminates the program, printing a failure message.

**NCARLB**

**ERPRT77** A portable FORTRAN 77 error handling package, adapted from the non-proprietary part of the PORT Mathematical Subroutine Library from Bell Labs.

**PORT**

**ENTER** Part of the PORT centralized error handling package that saves the current error recovery mode and storage allocation status. Used with LEAVE to bracket a sequence of Fortran statements, and cause the PORT storage allocation taking place within that sequence to be invisible to the outer part of the program.

**ENTSRC** Part of the PORT error handling package that tests and sets PORT error recovery mode. Saves the current recovery mode status and sets a new one. It also checks the error state, and if there is an active error state a message is printed. By convention, ENTSRC must be used upon entry to a subroutine which may return with a recoverable error, in order to guarantee that it is entered in a pristine error state.

**EPRINT** Part of the PORT centralized error handling package that prints the current error message if the program is in the error state; otherwise nothing is printed.

**ERROFF** Part of the PORT centralized error handling package that turns off the error state by setting the error number to zero.

**LEAVE** Part of the PORT centralized error handling package that restores prior error recovery mode and resets the stack. Can be used with the routine ENTER to
bracket a sequence of Fortran statements, and cause the storage allocation taking place within that sequence to be invisible to the outer part of the program.

NERERROR Part of the PORT centralized error handling package that returns the current error number (if any) or zero if the program is not in the error state.

RETSRC Part of the PORT centralized error handling package that sets the recovery mode to the status given by the input argument. A test is then made to see if a current error state exists which is unrecoverable. If so, an error message is printed and the run is terminated. By convention, RETSRC is used upon exit from a subroutine to restore the previous recovery mode status stored by the PORT routine ENTSRC.

SETEERR Part of the PORT centralized error handling package that sets the error indicator and, depending on whether the error is specified as recoverable or fatal and what the error recovery mode is, prints a message and provides a dump.

**SLATEC**

FDUMP Gives symbolic dump (modified locally).

NUMXER Returns most recent error number.

XERABT Aborts program execution and prints error message.

XERCLR Resets current error number to zero.

XERCTL Allows user control over handling of individual errors.

XERDMP Prints the error tables and then clears them.

XERMAX Sets maximum number of times any error message is to be printed.

XERROR Processes an error (diagnostic) message.

XERRWV Processes error message allowing 2 integer and two real values to be included in the message.

XGETF Returns current value of error control flag.

XGETUA Returns unit number(s) to which error messages are being sent.

XGETUN Returns the (first) output file to which messages are being sent.

XSEF Sets the error control flag.

XSETUA Sets up to 5 unit numbers to which messages are to be sent.

XSETUN Sets output file to which error messages are to be sent.

**SYSLIB**

RCVRBAD Skips to the first good data encountered.
Q6. Storage Allocation, Memory Management, and Overlays

**NCARLB**

**LCMREQ** LCMREQ is a collection of subroutines LCMREQ, LCMRD, and LCMWT for simulating LCM on the Cray. These routines are described in the NCAR System Library Routines Manual.

**PORT**

**ISTKGT** Part of the PORT dynamic storage allocation package that gets (allocates) an array from the storage stack.

**ISTKIN** Part of the PORT dynamic storage allocation package that is used to initialize the length of the dynamic storage stack.

**ISTKMD** Part of the PORT dynamic storage allocation package that changes the size of the last allocation made in the dynamic storage stack.

**ISTKQU** Part of the PORT dynamic storage allocation package that returns the amount of available space left in the stack.

**ISTKRL** Part of the PORT dynamic storage allocation package that releases the last storage allocation(s) requested.

**ISTKST** Part of the PORT dynamic storage allocation package that returns information on the status of the stack.

**SYSLIB**

**MEMAU** Puts the job in automatic field reduction mode.

**MEMFL** Allows the user to change the number of words of field length allocated to the job.

**MEMUC** Allows user to change the size of the code/data area (BA through HLM).

**ULFT** Allocates more user LFT space.

Q7. Job Control, Job Status, Operating System Requests

**NCARLB**

**JOBID** Returns sequence number, user name, user number, and project number in an array, left justified and blank filled.

**Q8QST4** Contains routines for gathering computing facility software usage statistics.
**SCILIB**

**WEOFSI**
Cray 1 routine for writing an /EOF between the CAL decks and the ENTRY decks of library $SCILIB. This allows $SCILIB to be assembled by a CAL directive followed by a CFT directive.

**SYSLIB**

**ACTTABLE**
Cray-1 routine for returning job accounting table (JTA).

**ALF**
Adds name to logical file table.

**CCS**
Control statement cracking routines CCS (callable from Fortran), $CS, and $CCS (callable from CAL only) that make the keywords and values on a control statement available for further processing.

**CHKEDRT**
Checks the range of the 'ED' and 'RT' parameters used typically in calls to PDM functions. Returns the binary conversion if in range.

**CRACK**
Reformats (cracks) a user-supplied string into verb, separators, keywords, and values. The cracked directive is placed in a user-supplied buffer.

**CSECHO**
Echoes a Cray-1 control statement to $LOG and $SYSTEMLOG, taking into consideration whether the job is interactive or batch.

**DRIVER**
Allows FORTRAN users to interface with the CAL driver macro.

**DRIVST**
Allows FORTRAN users to get an ASCII driver status.

**DSNDSP**
Contains $DSNDSP, callable only from CAL, which searches logical file table in user's I/O area for dataset name and returns DSP address.

**ECHO**
Contains FORTRAN and CAL callable routines which allows the user to change the suppression/allowance of classes of messages to the user logfile.

**EEXPR**
Evaluates an expression which has been cracked by the companion routine CEXPR.

**ENCRPT**
Encrypt the designated passwords using the keyword as by the index in the parameter list.

**ENDRPV**
Continues job step termination processing or clears an existing reprieve environment.

**EOF1**
This deck contains a *WEOF directive to write an end-of-file between CAL routines and CFT routines. This makes it possible to do a full mode UPDATE and get all CAL routines in the first file and all CFT routines in the second file. This assumes that the decks are placed and maintained in order by using the *MOVEDK UPDATE directive whenever new decks are added.

**ERECALL**
Allows the FORTRAN user to do everything the CAL user can do with the ERECALL macro.

**FINDPRV**
Finds the privilege code associated with a privilege name.

**FINDUSER**
Finds the user record and returns record number.

**FXPC**
Contains subprograms FXP (callable from FORTRAN) and $FXP (callable from CAL only) which format and write to a specified output dataset the contents of the exchange package (a block of memory associated with each Cray-1 program), the contents of the vector mask, and the return address.
FXPF | Prints exchange package.
GLPP | Returns lines from JCLPP.
GPARAM | A package of "get parameter" routines that process control statement parameter values from an already cracked control statement. $GP, $GPARAM, and $PAL are callable from CAL only. GETPARAM is callable from Fortran.
GTDSP | Contains $GTDSP (callable from CAL only) and GETDSP, which locate a Dataset Parameter Area, given a unit number or dataset name.
IJCOM | Allows FORTRAN users to implement inter-job communication.
INSASCI | Inserts ASCII parameters into a message.
ISP | Creates the initial connection between a Cray user job and an ISP subsystem. ISP statement processing will generate a taskinit message which will cause an initialization of a task on the ISP subsystem. This task will correspond to the Cray user job and provide the ISP functions requested by the job.
JNAME | FORTRAN-callable routine which gets the job name.
JSYMGET | Gets the value of a given JCL symbol.
JSYMSET | Sets the value of a given JCL symbol.
NORERUN | Controls monitoring of conditions causing job to be flagged as not rerunnable.
OPTION | Allows the user to set various user selectable parameters during job execution.
PDDMSG | Mechanism to request PDM message suppression.
PDMTA | Processes the track accesses parameter for permanent dataset related $SYSLIB routines.
PERF | It is used to interface with COS to obtain access to the hardware performance monitor. Various calls allow the gathering of processor utilization statistics and the subsequent reporting of these statistics to the calling routine.
PPL | Processes the keywords for a given directive after it has been cracked by the CRACK routine. Similar to GETPARAM, except that the directive may be obtained from some source other than $CS.
PRCW | Contains the subprogram $PRCW, callable from CAL only, which positions a dataset after the record control word.
RERUN | Allows the user to declare the job rerunnable.
SDSP | Contains programs $SDSP, $SLFT, and $ALF, which are callable only from CAL. $SDSP searches Dataset Parameter Area (DSP) for a dataset name and returns DSP address. $SLFT searches logical file table for dataset name and returns logical file table address. $ALF adds a name to the logical file table.
SETRPV | Transfers control to a specified routine upon encountering a user selected reprievable error condition.
SLFT | Searches LFT for dataset name and returns LFT address.
SUBMIT | Submits a job to Cray-1 input queue.
SYSREQ | Request the Cray-1 System to perform one of the F$XXX functions, and to return the status upon completion, since some functions have no-abort options.
SYSTEM | Makes a call to the Cray Operating System, executing the system function corresponding to a given function code, a list of which appears in the Cray-OS
Version 1 Reference Manual, Appendix C.

**VDSN**
Determine if a supplied name is a valid local dataset name; FORTRAN and CAL callable entries.

**XPFMT**
Formats exchange package.

**UTILITIES**

**BUILD**
Cray system control statement for generating and maintaining library datasets.

**CALL**
Cray system control statement for instructing the CRAY operating system to begin reading control statements from the first file of the indicated dataset.

**ECHO**
Cray system control statement for user control of messages written to user's logfile.

**ELSE**
Cray system control statement for use in a conditional control statement block.

**ELSEIF**
Cray system control statement for use in a conditional control statement block.

**ENDIF**
Cray system control statement for use in a conditional control statement block.

**ENDLOOP**
Cray system control statement for terminating an iterative control statement block.

**ENDPROC**
Cray system control statement for terminating an in-line procedure definition block.

**EXIT**
Cray system control statement indicating the point in the control statement file at which processing of control statements resumes following a job step abort from a program.

**EXITIF**
Cray system control statement defining conditions to skip remaining control statements in the conditional block.

**EXITLOOP**
Cray system control statement defining conditions under which the control statement block is to end.

**FLODUMP**
Cray system control statement for recovering and dumping flowtrace tables when a program aborts with flow tracing active.

**FTREF**
Cray system control statement for generating a listing containing information about a FORTRAN application.

**IF**
Cray system control statement for defining the beginning of a conditional block.

**IOAREA**
Cray system control statement for locking or unlocking that portion of the user field containing the user's Dataset Parameter Area and I/O buffers.

**ITEMIZE**
Cray system control statement for inspecting library datasets.

**JOB**
Cray system control statement defining the job to the operating system.

**LDR**
Cray system control statement for calling the loader into execution.

**LIBRARY**
Cray system control statement for specifying the library-defined dataset names that are to be searched during the processing of control statement verbs.

**LOOP**
Cray system control statement defining the beginning of an iterative block.

**MEMORY**
Cray system control statement for requesting a new field length and/or mode of field length reduction.

---

File Management

Category Q7
MODE  Cray system control statement for setting or clearing the Floating-point Error Interrupt flag in the mode register in the Exchange Package of the job.

MODIFY  Cray system control statement for modifying permanent dataset information established by the SAVE function or previously executed MODIFY function.

NORERUN  Cray system control statement specifying whether the operating system is to recognize flags that would make a job rerunnable.

OPTION  Cray system control statement for specifying user-defined options, such as the format of a job's listing.

PASCAL  Cray system control statement invoking the PASCAL compiler.

PERMIT  Cray system control statement for granting/denying specified users access to a permanent dataset.

PRINT  Cray system control statement for writing an expression value to the logfile in three formats: decimal integer, 22-digit octal, and ASCII string.

PROC  Cray system control statement defining the beginning of an in-line procedure definition block.

RELEASE  Cray system control statement for relinquishing access to the named datasets.

RERUN  Cray system control statement for unconditionally declaring a job to be rerunnable or nonrerunnable.

RETURN  Cray system control statement for returning control to the caller after reading JCL from a dataset.

REWIND  Cray system control statement for positioning the named datasets at the beginning of data.

ROLLJOB  Cray system control statement for protecting a job by writing it to disk for recovery if a system interruption occurs.

SAVE  Cray system control statement for making a local dataset permanent.

SET  Cray system control statement for changing the value of a specified JCL symbol.

SUBMIT  Cray system control statement for directing a job dataset to the CRAY input queue.

SWITCH  Cray system control statement for turning off/on pseudo sense switches.

SYSREF  Cray system control statement for reading special binary symbol tables written by CAL or APML, and producing a single cross-reference listing for the program modules represented in the tables.
S. Software Development Tools, Language Processors

**LOCLIB**

CONCORD  Source for Cray binary module CNCRD.

MORTMACROS  Data for the MORTRAN preprocessor that defines the Fortran extensions MORTRAN supports.

MORTRANCF  FORTRAN source for the MORTRAN preprocessor.

PFORTCF  Fortran source for PFORT on the Cray, a Fortran '66 standards checker that also checks for some kinds of errors that compilers can't catch.

**SYSLIB**

LOADGO  Loads and executes a user absolute binary from a local dataset which contains the binary image.

**UTILITIES**

CAL  Cray system control statement for initiating the loading and execution of the CAL assembler.

CFT  Cray system control statement for initiating the loading and execution of the FORTRAN compiler.

CFTxxx  Locally developed Cray command for accessing libraries and system texts necessary for use of edition xxx of the CFT, where xxx is the three-digit number associated with a compiler release. For example, CFT114 accesses the libraries and texts associated with release 1.14 of CFT.

IFTRAN  A preprocessor for converting IFTRAN code to FORTRAN.

LDR  Cray system control statement for calling the loader into execution.

MACRO  VOS Software Tools command for general-purpose macro processing.

MORTRAN  Externally acquired preprocessor for converting MORTRAN code to FORTRAN.

PASCAL  Cray system control statement invoking the PASCAL compiler.

PFORT  Externally acquired Cray command for checking FORTRAN for divergence from ANSI '66.

RATFOR  VOS Software Tools command for Ratfor preprocessing. Ratfor (“rational Fortran”) is a structured Fortran preprocessor.

REDUCE  A symbolic algebra system (available only on the IBM 4341) that provides a language for general algebraic computations. Its capabilities include expansion and ordering of polynomials and rational functions, symbolic differentiation, symbolic integration, simplification of expressions, and calculations with symbolic matrices.

ROFF  VOS SOFTWARE TOOLS text-formatting utility.
SEGLDR: Cray system control statement for invoking the automatic loader.
AMOSLIB

ORIGIN AMOSLIB is an extensive collection of over 61 special function routines developed by D.E. Amos. NCAR acquired the collection from Sandia Laboratories and made it available to its Cray users in July 1979. Prior to that, it had been available on NCAR’s CDC 7600 computer. AMOSLIB includes routines for calculating error functions, gamma functions, beta functions, Bessel functions, Airy functions, sine and cosine integrals, and various distribution functions.

DOCUMENTS The Consulting Office has a copy of the Sandia Laboratories publication

AMOSLIB, A Special Function Library
Version 9/77
SAND 77-1390

which contains writeups for each routine in AMOSLIB. This publication is also available from the National Technical Information Service. On-line writeups are included as the first block of comments in each AMOSLIB source file. The comment block write-up at the beginning of each file may be listed using the GETDOC utility on the Cray or IBM 4341. For example, to list the documentation for BESI, use the following command in a Cray job:

GETDOC,LIB=AMOSLIB,DOC=BESI.

ACCESS AMOSLIB is available as an object library on the Cray. To access the Cray object library, include AMOSLIB in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=AMOSLIB.

would load any needed routines from AMOSLIB after loading from the default libraries.

To access AMOSLIB source files from the Cray, use the GETSRC command. For example, to place the source for BESI and GAMLN on Cray local dataset SRC, include the following commands in a Cray job:

GETSRC,LIB=AMOSLIB,FILE=(BESI:GAMLN),L-SRC.
AMOSLIB

SUPPORT  AMOSLIB is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any author- or vendor-supplied updates. Documentation and usage statistics are maintained for AMOSLIB. It is not guaranteed that AMOSLIB will be supported on future computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. The simple way in which AMOSLIB was adapted to the Cray means that it does not take advantage of the full range of floating point numbers on the Cray. No attempts have been made to exploit the vectorization capabilities of the Cray in AMOSLIB.
EDA

ORIGIN  EDA is a collection of FORTRAN subroutines for exploratory data analysis (EDA) which NCAR installed in July 1985. The purpose of these routines and their calling sequence are described in “Applications, Basics, and Computing of Exploratory Data Analysis” by Paul F. Velleman and David C. Hoaglin, Duxbury Press, 1981.

John Tukey has been a leading figure in promoting the use of EDA techniques, particularly with the publication of his book, “Exploratory Data Analysis”, Addison-Wesley, 1977. EDA emphasizes the discovery of the structure and anomalies in a set of data. It also emphasizes minimizing the number of assumptions made about the data (such as normality, additivity of effects, and independence of observations).

The EDA subroutines described in the book by Velleman and Hoaglin include stem-and-leaf displays, letter-value displays, boxplots, x-y plots, resistant line fitting, data smoothers, coded tables, median polish, and rootograms. Calling sequences are documented at the end of the chapter describing the technique. The graphics produced by these routines are simple line-printer plots.

DOCUMENTS  The EDA routines are documented in “Applications, Basics, and Computing of Exploratory Data Analysis” by Paul F. Velleman and David C. Hoaglin, Duxbury Press, 1981, which is available in the NCAR Mesa Library and the SCD Computing Library.

ACCESS  EDA is available as an object library on the Cray-1’s. To access the Cray-1 object library, include EDA in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=EDA.

will load any needed routines from EDA after loading from the default libraries.

To access EDA source files from the Cray-1, use the GETSRC command. The EDA source is stored as a single file, which is about 3000 lines long.

The subroutine CINIT (see page 309 of the book by Velleman and Hoaglin) must be called before any of the EDA routines are called. The constants IEPSI (machine epsilon) and IMAXIN (maximum integer) are input parameters for CINIT and may be set with the following system functions:

\[
\begin{align*}
\text{IEPSI} & = \text{R1MACH (3)} \\
\text{IMAXIN} & = \text{I1MACH (9)}
\end{align*}
\]
EDA

SUPPORT EDA is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any author- or vendor-supplied updates. Documentation and usage statistics are maintained for EDA. It is not guaranteed that EDA will be supported on future computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation.
EISPACK

ORIGIN

EISPACK is a collection of subroutines for solving eigenvalue and eigenvector problems. It was developed as part of the NATS project (National Activity to Test Software), of which NCAR was a test site, and acquired by NCAR in June 1974. The version supported at NCAR was originally certified for use on Control Data 6000/7000 computers, but has been made available on the Cray as well. Several drivers that call the correct sequence of EISPACK routines to solve common eigenvalue problems are included in LOCLIB. Version 3.0 of EISPACK is currently installed on the NCAR Cray computers.

EISPACK version 2.0 has been made available on the Cray supported library $SCILIB. On the Cray EISPACK routines should be accessed from $SCILIB, when possible, since these routines have been optimized for the Cray by various changes to the FORTRAN source to enhance vectorization. Access instructions are provided below.

DOCUMENTS

Information about how to solve eigenproblems with the EISPACK routines is available in the Consulting Office in the book

Matrix Eigensystem Routines--EISPACK Guide Extension
by B.S. Garbow, J.M. Boyle, J.J. Dongarra, and C.B. Moler
Lecture Notes in Computer Science, Vol. 51 (1977)
Springer-Verlag, New York, N.Y., 343 pages.

which contains writeups for each routine in EISPACK. On-line documentation is available for all EISPACK files, and may be listed by using GETDOC on the Cray. For example, to list the documentation for TINVIT and COMBAK from the Cray, include the following command in a Cray job:

GETDOC,LIB=EISPACK,DOC=(TINVIT:COMBAK).

ACCESS

EISPACK is available as an object library on the Cray and as part of $SCILIB in the Cray.

To access any of the EISPACK routines that are in $SCILIB, no explicit access is necessary, since $SCILIB is searched by default. Routines that are in EISPACK 3.0 but not in $SCILIB (for example RSM) must be accessed from the EISPACK object library. To make sure users get the benefits of $SCILIB optimized routines, even when accessing EISPACK routines not included in $SCILIB, all of the routines included in $SCILIB have been removed from the EISPACK object library on the Cray.

LDR,LIB=EISPACK.

will cause the loader to take whatever optimized versions exist from $SCILIB, and load any other EISPACK routines from the EISPACK
To access EISPACK source files from the Cray, use the GETSRC command. For example, to place the source for BALANC and ELMHES on CRAY local dataset SRC, include the following command in a Cray job:

GETSRC,LIB=EISPACK,FILE=(BALANC:ELMHES),L=SRC.

SUPPORT

EISPACK is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for EISPACK. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. The simple way in which EISPACK was adapted to the Cray means that it does not take advantage of the full range of floating point numbers on the Cray. No attempts have been made to exploit the vectorization capabilities of the Cray in EISPACK. Questions about the algorithms used or corrections to the routines should be directed to the subroutine authors, whose addresses are listed in the on-line documentation.
EISPKD

ORIGIN EISPKD is collection of FORTRAN subroutines that compute the eigenvalues and eigenvectors for different classes of double precision matrices. This package was developed at Argonne National Laboratory and represents the finest quality software for solving eigenvalue problems. Until the current release (EISPACK 3.0) there has been no official double precision version. EISPKD is machine independent double precision version of EISPACK. It is a collection of 75 FORTRAN subprograms.

DOCUMENTS Information about how to solve eigenproblems with the EISPACK routines is available in the Consulting Office in the book

Matrix Eigensystem Routines--EISPACK Guide Extension
by B.S. Garbow, J.M. Boyle, J.J. Dongarra, and C.B. Moler
Lecture Notes in Computer Science, Vol. 51 (1977)
Springer-Verlag, New York, N.Y., 343 pages.

which contains writeups for each routine in EISPACK. On-line documentation is available for all EISPKD files, and may be listed by using GETDOC on the Cray. For example, to list the documentation for TINVIT and COMBAK from the Cray, include the following command in a Cray job:

GETDOC,LIB=EISPKD,DOC=(TINVIT:COMBAK).

ACCESS EISPKD is available as an object library on the Cray. To access the Cray object library, include EISPKD in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=EISPKD.

To access EISPKD source files from the Cray, use the GETSRC command. For example, to place the source for BALANC and ELMHES in Cray local dataset SRC, include the following command in a Cray job:

GETSRC,LIB=EISPKD,FILE=(BALANC:ELMHES),L=SRC.

SUPPORT EISPKD is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for EISPKD. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local exper-
tise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in EISPKD. Questions about the algorithms used or corrections to the routines should be directed to the subroutine authors, whose addresses are listed in the on-line documentation.
FITPACK

ORIGIN  FITPACK is a package of FORTRAN subroutines for curve and surface fitting employing splines under tension (a generalization of cubic splines), acquired from the University of Texas at Austin by NCAR in April 1982. FITPACK has been incorporated into an object library named FITPACK on the Cray.

FITPACK contains over 58 user-level FORTRAN routines for interpolation, differentiation, integration, and smoothing using splines under tension for data that represents the following kinds of mappings:

- from a one dimensional space into a one dimensional space, that is a sequence of functional values is given to represent a function of one variable;
- from a one dimensional space into a two dimensional space, that is a sequence of coordinate pairs is given to represent a curve in the plane;
- from a one dimensional space into a three dimensional space, that is a sequence of coordinate triples is given to represent a curve in space;
- from a two dimensional space into a one dimensional space, that is a sequence of functional values is given at points on a rectangular grid on a surface to represent a scalar function defined on a surface;
- from a three dimensional space into a one dimensional space, that is a sequence of functional values is given at points on a three dimensional rectangular grid to represent a scalar field;
- from a two dimensional space into a three dimensional space, that is a rectangular grid of coordinate triples is given to represent a surface in space.

DOCUMENTS  A guide to FITPACK with brief descriptions of each subprogram and a set of charts for selecting major software paths is available in the Consulting Office library. On-line documentation for FITPACK subprograms is also available on the Cray. For example to access the documentation for the subprograms CURV1, CURVS and ZURFN1 include the following command in a Cray job:

GETDOC,LIB=FITPACK,DOC=(CURV1:CURVS:ZURFN1).
FITPACK

ACCESS
FITPACK is available as an object library on the Cray. To access the Cray library, include FITPACK in the list of parameter values for the LIB keyword on the LDR statement. For example:

    LDR,LIB=FITPACK.

The FORTRAN source code for FITPACK is available on the CRAY-1 via the GETSRC statement. For example, command

    GETSRC,LIB=FITPACK,FILE=(CURV1:CURV2),L=SRC.

will place the two requested files on Cray local dataset SRC.

SUPPORT
FITPACK is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in FITPACK. Questions about the algorithms used or corrections to the routines should be directed to the author, Alan K. Cline of the Department of Computer Science, University of Texas at Austin.
FUNPACK

ORIGIN  FUNPACK is a collection of subroutines for calculating Bessel functions, elliptic integrals, exponential integrals, and a few other special functions. It was developed as part of the NATS project (National Activity to Test Software), of which NCAR was a test site. The FUNPACK software was acquired by NCAR in 1976. The routines in NCAR's version of FUNPACK contain well-tested high-accuracy approximations specifically tailored for the Control Data 6000/7000 computers. The Cray version of FUNPACK was developed at Boeing Computer Services by modifying the original CDC version.

DOCUMENTS  There is no manual for the routines in FUNPACK. Information about the routines and how to use them may be found in on-line documentation available from the Cray.

For example, to list the documentation for BESYN and MONERR from the Cray, include the following command in a Cray job:

```
GETDOC,LIB=FUNPACK,DOC=(BESYN:MONERR).
```

ACCESS  FUNPACK is available as an object library and source library on the Cray.

To access the Cray object library, include FUNPACK in the list of parameter values for the LIB keyword on the LDR statement. For example

```
LDR,LIB=FUNPACK.
```

would load any needed routines from FUNPACK after loading from the default libraries.

To access FUNPACK source files from the Cray, use the GETSRC command. For example, to place the source for BESYN and MONERR on Cray local dataset SRC, include the following command in a Cray job:

```
GETSRC,LIB=FUNPACK,FILE=(BESYN:MONERR),L-SRC.
```

SUPPORT  FUNPACK is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for FUNPACK. The NCAR Libraries Group does not guarantee that the software will
survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in FUNPACK. The simple way in which FUNPACK was adapted for the Cray does not take advantage of the full range of floating point numbers on the Cray. Questions about the algorithms used or corrections to the routines for the Control Data version of FUNPACK should be directed to the author,

W. J. Cody
Argonne National Laboratory
Applied Mathematics Division
Argonne, IL 60439
IMSL

ORIGIN

IMSL (International Mathematical and Statistical Libraries) is an extensive commercial library leased by NCAR for use on its Cray computers. Its availability at NCAR was announced in June 1972. IMSL routines are generally well documented and of high quality. This library is proprietary, so users may not copy these routines, transport them to other institutions, or use them on any machines other than NCAR's Cray machines. Versions of IMSL are available at most computing installations where scientific research is supported. A new edition of IMSL is released approximately once a year. Version 9.2 of IMSL is currently installed on the NCAR Cray computers.

DOCUMENTS

A copy of the IMSL Reference Manual containing write-ups of all the IMSL routines is available for reference in the Consulting Office. The manual also includes informative chapter introductions that offer an overview of the capabilities in each problem area and guide the user to the best choice of IMSL routine for a particular problem. Additional copies of the IMSL Reference Manual may be purchased from:

IMSL
Sixth Floor--NBC Building
7500 Bellaire Boulevard
Houston, Texas 77036

Additionally, each IMSL source file includes a write-up in comment cards at the beginning of the routine. These on-line write-ups may be listed using the GETDOC utility. For example, to list the write-ups for ABIBN and NDKER, run a Cray job that includes the following:

GETDOC,LIB=IMSL,DOC=(ABIBN:NDKER).

Some IMSL routine names are not the same as the name of the corresponding source file. In these cases, the notation F=R is used in the short descriptions to mean that the source and documentation for routine R is stored in a source file named F. For example, MERF=ERF implies that the routine name is ERF and that the source and documentation are in file MERF.

To supplement the on-line documentation for IMSL routines and to provide simple tests that verify that a routine correctly solves a sample problem, a test driver is available for each IMSL package. To see the results of the example test run, use the TEST utility on the Cray. For example, to run the sample program that calls ABIBN, include the following command in a Cray job:

TEST,LIB=IMSL,FILE=ABIBN.
IMSL

ACCESS
IMSL is available as an object library and a source library on the Cray.

To access the Cray IMSL object library, include IMSLIB in the list of parameter values for the LIB keyword on the LDR statement. For example,

LDR,LIB=IMSLIB.

loads any needed routines from IMSL after loading from the default libraries.

To access source files on the Cray, specify IMSL for the LIB parameter in the GETSRC command. For example, to write the source for ABIBN and NDKER as two files on local dataset SRC, include the following statement in a Cray job:

GETSRC,LIB=IMSL,FILE=(ABIBN:NDKER),L-SRC.

SUPPORT
IMSL is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. The IMSL library is accompanied by a test suite which is also available to Cray users. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation.

Questions about the clarity of the documentation, possible IMSL Library related programming problems, choice of IMSL routines for a particular problem, or planned IMSL Library evolution may be directed to the code designers or programmers by writing IMSL or calling (713)772-1927. Prior to contacting IMSL, users are encouraged to bring programming and library access questions to the NCAR Consulting Office.
ITPACK

ORIGIN

ITPACK is a package of FORTRAN subroutines, developed at the University of Texas at Austin, for solving large sparse linear systems of the form $Ax=b$. NCAR acquired the package for its Cray computer users in February 1980. Version 2C of ITPACK is currently installed on the NCAR's Cray machines.

ITPACK contains seven adaptive accelerated iterative procedures based on the Jacobi, successive over-relaxation (SOR), symmetric SOR (SSOR), and reduced system (RS) methods. With the exception of SOR the convergence of these basic methods has been accelerated by semi-iteration (Chebyshev acceleration) or conjugate gradient acceleration. All seven methods are available with adaptive parameter estimation and automatic stopping tests.

DOCUMENTS

For a discussion of the methods used in ITPACK see references [1] and [2]. A copy of the ITPACK 2C User's Guide is available for reference in the Consulting Office. Users are cautioned to use current documentation because the present version of ITPACK differs from its predecessors. On-line documentation is also available by using the Cray command GETDOC, as in:

GETDOC,LIB=ITPACK,DOC=ITPACKDC.

This documentation includes a definition of all of the subroutine calling sequences, a discussion of the sparse data structure required by all of the ITPACK routines and a description of some other user level routines which are included in the package to facilitate setting up the sparse data structure.

To list all subroutine descriptions including ITPACK under the categories "F4a4. Linear Systems: Real Non-Symmetric Sparse" and "F4b4. Linear Systems: Real Symmetric Sparse", run a Cray job that includes the following command:

GETDOC,LIB=DOCLIB,DOC=(F4A4:F4B4).

To list the on-line writeup of a routine of ITPACK routine JCG, for example, run a Cray job that includes the following command:

GETDOC,LIB=ITPACK,DOC=JCG.

To supplement the on-line documentation for ITPACK routines and to provide simple tests that verify that a routine correctly solves a sample problem, a test driver is available for each ITPACK package. To see the results of the example test run, use the TEST utility on the Cray. For example, to run the sample program that calls JCG, include the following command in a Cray job:
ITPACK

TEST,LIB=ITPACK,FILE=JCG.

ACCESS ITPACK is available as an object library and a source library on the Cray. To access the Cray object library, include ITPACK in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=ITPACK.

would load any needed routines from ITPACK after loading from the default libraries.

To access the FORTRAN source for the ITPACK library from the Cray use the GETSRC utility. For example to place the source file for the subprogram JCG on the Cray local dataset SRC, run a Cray job that includes the following command:

GETSRC,LIB=ITPACK,FILE=JCG,L=SRC.

SUPPORT ITPACK is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for ITPACK. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in ITPACK. Questions about the algorithms used or corrections to the routines should be directed to the subroutine authors, whose addresses are listed in the on-line documentation.

REFERENCES


[3] Grimes, R., Kincaid, D., and Young, D. ITPACK 2.0 user's guide. CNA-150, Center for Numerical Analysis, Univ. of Texas, Austin, Aug. 1979. (Users should note subroutine interface changes between this version and current version.)

LINPACK

ORIGIN LINPACK is a collection of Fortran programs for solving various types of linear systems. It was acquired by NCAR in 1978. The routines are designed to be completely machine-independent and to run at optimum efficiency in most operating environments. LINPACK deals with general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square matrices, as well as with least squares problems and the QR and singular value decompositions of rectangular matrices. At NCAR, LINPACK is supported on the Cray.

There are four versions of most LINPACK subroutines: single precision, double precision, complex, and double precision complex. The initial letter of the subroutines for each of these types is S, D, C, or Z, respectively. None of the double precision complex routines that begin with a Z are supported at NCAR.

Each of the single precision and complex LINPACK routines have been optimized for the Cray and included on $SCILIB$, the library of scientific and mathematical software maintained by Cray Research. $SCILIB$ is searched by default, so no specific access is needed on the Cray to use the single precision or complex LINPACK routines. To make sure users get the benefits of these optimized routines, even when accessing double precision routines from the LINPACK library, all of the single precision and complex routines have been removed from the LINPACK object library on the Cray.

DOCUMENTS Information about how to solve linear systems with the LINPACK routines is available in the Consulting Office in the book

LINPACK Users' Guide
by J.J. Dongarra, C.B. Moler, J.R. Bunch, and G.W. Stewart
SIAM, Philadelphia, 1979

which contains writeups for each routine in LINPACK, as well as some general advice about linear systems. On-line writeups are available in comment card form for all LINPACK files. These write-ups may be listed by using GETDOC on the Cray.

For example, to list the writeups for SGECO and SGESL from the Cray, run a job that includes the following command:

GETDOC,LIB=LINPACK,DOC=(SGECO:SGESL).
LINPACK is available as object and source libraries on the Cray.

To access the Cray-optimized LINPACK routines no explicit access is needed, since such routines will be loaded from $SCILIB by default if any of them have been called. To access the Cray object library for LINPACK routines not on $SCILIB, include LINPACK in the list of parameter values for the LIB keyword on the LDR statement. For example

```
LDR,LIB=LINPACK.
```

would load any needed routines from LINPACK after loading from the default libraries.

To access LINPACK source files from the Cray, use the GETSRC command. For example, to place the source for SGECO and SGEFA on Cray local dataset SRC, include the following commands in a Cray job:

```
GETSRC,LIB=LINPACK,FILE=(SGECO;SGEFA),L=SRC.
```

LINPACK is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for LINPACK. The LINPACK project supports the routines in LINPACK in the sense that detailed information on the testing procedures is available, and reports of poor or incorrect performance in conventional Fortran operating environments will gain immediate attention from the developers. Reports of suspected errors, descriptions of interesting applications and other comments may be sent to:

```
LINPACK Project
Applied Mathematics Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
```
LOCLIB

ORIGIN  LOCLIB is a source library of locally developed and acquired general-purpose mathematical and utility software. LOCLIB consists of software on local NCAR libraries previously known as ULIB, XLIB, and CRAYLIB, as well as routines acquired since ULIB, XLIB, and CRAYLIB were merged into one library. LOCLIB is most noted for its locally developed graphics and PDE packages. Many of the packages on LOCLIB have been incorporated into the Cray object library $NCARLB. FORTRAN source files and library data files reside on LOCLIB. The majority of packages on LOCLIB were written at NCAR, although most of the recent packages added to LOCLIB were acquired from other sources and enhanced to conform to NCAR library conventions.

DOCUMENTS  No separate manual for LOCLIB exists, although the graphics utilities on LOCLIB are documented in the NCAR Graphics Software manual, available from the Consulting Office. On-line writeups are included as comments in each source file.

The comment write-up at the beginning of each file may be listed using the GETDOC utility on the Cray. For example, to list the writeup for ADQUAD, include the following command in a Cray job:

GETDOC,LIB=LOCLIB,DOC=ADQUAD.

To supplement the on-line documentation for LOCLIB routines and to provide simple tests that verify that a routine correctly solves a sample problem, a test driver is available for most LOCLIB packages. To see the results of the example test run, use the TEST utility on the Cray. For example, to run the sample program that calls ADQUAD, include the following command in a Cray job:

TEST,LIB=LOCLIB,FILE=ADQUAD.

ACCESS  Many of the LOCLIB routines are maintained on the Cray binary library $NCARLB. Accessing these routines requires nothing more than calling the routines and invoking the loader. However, some LOCLIB routines cannot be placed in $NCARLB because of entry name conflicts.

To access LOCLIB source files from the Cray, use the GETSRC command. For example, to place the sources for ODEN and CONRAN on CRAY dataset SRC, include the following command in a Cray job:

GETSRC,LIB=LOCLIB,FILE=(ODEN:CONRAN),L=SRC.
SUPPORT  The graphics utilities on LOCLIB are fully supported at NCAR. Other LOCLIB routines are supported at a lower level, since many of them have been superceded by superior packages on other acquired libraries. The Software and Libraries Group is responsible for correcting errors in LOCLIB routines, and ensuring that LOCLIB routines operate correctly under new operating systems or on new computers. If users discover errors or inadequacies in LOCLIB routines, they are urged to bring them to the attention of the Consulting Office or a member of the Libraries Group. An effort will be made to diagnose and correct errors in LOCLIB routines or their documentation. Modifications and deletions of LOCLIB files will be done with as little disturbance to users as is practical.
MINPACK

ORIGIN

MINPACK-1 is a package of FORTRAN subroutines for solving systems of nonlinear equations and nonlinear least squares problems. The MINPACK-1 software was produced at Argonne National Laboratory as part of a research effort whose goal is the development of a systematized collection of quality optimization software. It was acquired by NCAR in March, 1981. The programs in MINPACK-1 were designed to be reliable, easy to use, and transportable. The single-precision version of MINPACK-1 has been incorporated into an object library named MINPACK on NCAR's Cray. Future additions to MINPACK will include software for unconstrained minimization and constrained optimization.

For each problem area covered by MINPACK, an easy-to-use driver that assumes default values for certain parameters and provides a limited amount of information about the solution is available, as well as a core routine which provides more flexibility. The user can decide whether to provide the Jacobian matrix of partial derivatives, or to let the package calculate and use a finite difference approximation to the Jacobian. The main advantage to providing the Jacobian is increased reliability in some cases, for example when the function values are subject to errors. The coding of a routine to compute the Jacobian can be an error-prone task, so an entry is provided to check that the Jacobian is consistent with the function values.

DOCUMENTS

Information about how to solve nonlinear systems of equations and nonlinear least squares problems with the MINPACK routines is available in the Consulting Office Library in the manual

User Guide for MINPACK-1
by Jorge J. More, Burton S. Garbow, and Kenneth E. Hillstrom
Argonne National Laboratory Report ANL-80-74

Copies of this manual may be ordered from

National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes: A12 (printed copy)
A01 (microfiche)

On-line documentation is contained as a block of comment cards in each user-level subroutine in MINPACK. These files may be listed by using the GETDOC command on the Cray.

For example, to list the writeups for HYBRD1 and CHKDER from the Cray, run a job that includes the following command:

Libraries 326
ACCESS To access the Cray object library, include MINPACK in the list of parameter values for the LIB keyword on the LDR statement. For example

\[
\text{LDR,LIB=MINPACK.}
\]

would load any needed routines from MINPACK after loading from the default libraries.

The FORTRAN source code for MINPACK is available on the Cray via the GETSRC statement. For example, the command

\[
\text{GETSRC,LIB=MINPACK,FILE=(CHKDER:DOLEG),L=SRC.}
\]

will place the two requested files on Cray local dataset SRC.

SUPPORT MINPACK is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for MINPACK. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in MINPACK. Questions about the algorithms used or corrections to the routines should be directed to the subroutine authors, whose addresses are listed in the on-line documentation.
NAG

ORIGIN

NAG is a comprehensive library of algorithms for the solution of numerical problems on computers. It was leased from the Numerical Algorithms Group in April 1982 for use on NCAR’s Cray computer, and later the lease was expanded to include NCAR’s second Cray.

The NAG Library was originally developed as a collaborative effort among a small number of British computer centers, but is now supported by a large number of specialists in numerical analysis and applications programming from all over the world. Inclusion of software in this library is based on usefulness, robustness, numerical stability, accuracy, and speed. Attractive features of the NAG Library include the state-of-the-art numerical algorithms that it implements, careful error checking, clear documentation, and helpful chapter introductions which serve as overviews of problem areas as well as guides to the selection of routines.

A new version of NAG is released approximately once a year. Mark 11 is currently installed on the NCAR Cray computers. This version includes some optimizations specifically intended to enhance the performance of linear algebra routines on vector processors like the Cray computers.

DOCUMENTS

A copy of the six-volume NAG Library Manual containing write-ups of all the NAG routines is available for reference in the NCAR Consulting Office Library. The manual includes informative chapter introductions that offer an overview of the capabilities in each problem area and guide the user to the best choice of NAG routine to use for a particular problem. A smaller NAG Mini Manual that includes all of these chapter introductions and other essential information needed to select appropriate routines and use the NAG library effectively is also available in the Consulting Office Library. The Mini Manual or the full NAG Library Manual may be purchased from NAG. Copies may be ordered from

NAG (USA) Inc.
1131 Warren Avenue
Downers Grove, Illinois 60515

phone: (312) 971-2337

NCAR leases on-line supplementary documentation for the NAG FORTRAN Library. To list all of the brief summaries of routine names and purposes provided by NAG (about 15 pages), run a Cray job that includes the following command:

GETDOC,LIB=NAG,DOC=SUMMARY.

To print a list of keywords and the NAG routine name suffixes that are associated with each keyword, run a Cray job that includes the

Libraries 328
following command:

GETDOC,LIB=NAG,DOC=KEYWORDS.

To list the on-line write-up for a NAG routine, for example the routine named D02SAF, run a Cray job that includes the following command:

GETDOC,LIB=NAG,DOC=D02SAF.

To supplement the on-line documentation for NAG routines and to provide simple tests that verify that a routine correctly solves a sample problem, a test driver is available for each NAG package. To see the results of the example test run, use the TEST utility on the Cray. For example, to run the sample program that calls D02SAF, include the following command in a Cray job:

TEST,LIB=NAG,FILE=D02SAF.

ACCESS  NAG is available as an object library on the Cray. The FORTRAN source for NAG is proprietary.

To access the Cray object library, include NAG in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=NAG.

will load any needed routines from NAG after loading from the default libraries.

SUPPORT  NAG is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation is maintained for NAG software. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation.
$NCARLB is a locally maintained Cray-1 object library made from portions of the source library LOCLIB (formerly CRAYLIB, ULIB, and XLIB). Its existence was announced at NCAR in October, 1978, and its development continues at present. Some LOCLIB packages are not included on $NCARLB because of entry name conflicts.

No separate documentation for $NCARLB exists. Write-ups for $NCARLB routines appear in comment blocks in the source files on LOCLIB.

Routines in $NCARLB are accessed by calling them in FORTRAN or CAL programs. $NCARLB is searched by default by the Cray loader, so no explicit mention of $NCARLB on the loader statement is needed. To access the source for a routine on $NCARLB requires knowing where the source resides. A table containing this information is kept in hard-copy copies of the NCAR Software Catalog.

The level of support varies for various routines in $NCARLB. Graphics utilities from LOCLIB sources are fully supported. The Libraries Group takes responsibility for correcting, updating, and testing these routines, to ensure that they survive operating system changes. Other $NCARLB routines receive only limited support. Responsibility for consulting about, testing, and modifying these routines is assumed by the authors or local implementors.
PORT is a general-purpose mathematical subroutine library developed by Bell Laboratories and purchased from Western Electric for NCAR's Cray computers. Its availability to NCAR users was announced in June 1979. The most outstanding features of the PORT library are its portability, its emphasis on ease of use, careful error checking and clear documentation. The structure of the library allows calling sequences that need not include error flags or workspace arguments, because PORT uses a centralized error handling facility and portable dynamic workspace allocation from a stack. The selection of numerical algorithms and their implementations in PORT have been governed by considerations of portability, robustness, accuracy, and efficiency.

A copy of the PORT reference manual containing write-ups of all the PORT routines is available for reference in the Consulting Office. In addition, the NCAR Computing Facility has copies of the PORT manual available for users. The manual includes informative chapter introductions that offer an overview of the capabilities in each problem area and guide the user to the best choice of PORT routine for a particular problem. There is currently no on-line documentation available for PORT.

PORT is available as an object library on the Cray. The Fortran source for PORT is proprietary.

To access the Cray object library, include PORT in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=PORT,

would load any needed routines from PORT after loading from the default libraries.

PORT is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for PORT. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization
capabilities of the Cray in PORT.
SAS

SAS (Statistical Analysis System) is a large commercial package leased by NCAR for use on the IBM 4341 from SAS Institute Inc. NCAR began leasing base SAS and SAS/ETS (Economic and Time Series Library) in October 1982. In January 1986 version 5 of SAS was installed. The current version is printed in the upper right-hand corner of each SAS run.

SAS is a software system for data management, statistical analysis, and report writing. It contains a high level programming language and a library of over 100 data management and statistical facilities. SAS also has macro facilities and the ability to incorporate a user supplied FORTRAN program.

SAS statistical procedures described in this catalog include:

- analysis of variance and covariance
- regression with autocorrelated errors
- cluster analysis
- correlation analysis (Pearson, Spearman, and Kendall)
- discriminant analysis
- factor analysis
- frequency and crosstabulation tables
- multiple linear regression (stepwise and all possible subsets)
- matrix manipulation
- non-linear regression
- probit analysis
- spectral analysis
- log-linear modeling
- canonical correlation analysis
- principal components analysis
- nonparametric analysis
- response surface estimation
- survival analysis

DOCUMENTS

A copy of An Introduction to Using SAS on NCAR'S IBM 4341 Computers is available in the SCD Computing Library. Some of the SAS manuals described in this introduction are available for reference use in the SCD Computing Library, for check-out from the NCAR Mesa Library, or may be purchased from:
SAS Institute, Inc.
P.O. Box 8000
Cary, North Carolina 27511-8000

Phone: (919) 467-8000

No on-line documentation is available, however, a help facility is available to full-screen interactive users.

**ACCESS**
SAS may be accessed on the IBM 4341 in noninteractive mode by typing in

```
SAS [filename]
```

where [filename] is the name of the file with SAS commands in it.

To access SAS in interactive, line-mode type in:

```
SAS (NODMS
```

To access SAS in interactive, full-screen mode type in:

```
SAS
```

**SUPPORT**
SAS is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any author- or vendor-supplied updates. It is not guaranteed that SAS will be supported on future computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation.
$SCILIB is a library of scientific applications programs written to run optimally on the Cray and maintained by Cray Research, Inc. The current version of $SCILIB includes subprograms for linear algebra, fast Fourier transforms, digital filters, packing/unpacking of integers and gather/scatter vector operations. Replacing inner loops with calls to appropriate $SCILIB routines often results in significantly faster execution times on the Cray.

$SCILIB routines are documented in

- Cray-1 Library Reference Manual
- CRI Publication SR-0014

The comment-block write-up at the beginning of each file may be listed using the GETDOC utility on the Cray. For example, to list the on-line write-up for MXMA, include the following command in a Cray job:

```
GETDOC,LIB=$SCILIB,DOC=MXMA.
```

The name used for a $SCILIB routine in the NCAR Library Catalog is the name of the source UPDATE-deck name. This is usually the same as the name of the user-callable subroutine, but may be different.

Routines in $SCILIB are accessed automatically by the Cray loader. These may be accessed and listed by using the GETSRC command. For example, to place the source for FILTERG in Cray local dataset SRC, include the following command in a Cray job:

```
GETSRC,LIB=$SCILIB,FILE=FILTERG,L=SRC.
```

$SCILIB is a fully supported Cray product. Problems with $SCILIB routines should be brought to the attention of the Consulting Office. A consultant will verify the problem and report it to the Cray site analyst.
SLATEC

ORIGIN

SLATEC (Sandia Laboratory, Los Alamos National Laboratory, Air Force Weapons Laboratory Technical Exchange Committee) common mathematical library is a collection of FORTRAN routines for scientific calculations. It resulted from an experiment in resource sharing by the computing departments of Energy Laboratories and other affiliated organizations with the objective to cooperatively assemble and install at each site a mathematical subroutine library characterized by portability, good numerical technology, good documentation, robustness and quality assurance. Thus, programs transported from one site to another have access to a common set of mathematical routines via the SLATEC library. The effort began in 1977, with the library being installed at all sites in 1981. Its availability was announced to NCAR users in May, 1983. The library is designed by users with users in mind. For example, the library includes user-oriented “drivers” to solve problems in linear systems and eigenanalysis by a single subroutine call using routines from LINPACK or EISPACK. These drivers provide for argument checking and error reporting.

SLATEC software is in the public domain, with no proprietary restrictions, hence it can be copied or incorporated in user’s programs to be taken elsewhere.

The SLATEC library has more than 594 user-level routines. There is much overlap between the SLATEC library and other Cray libraries such as LINPACK, EISPACK, FUNPACK, AMOSLIB and $SCILIB. Version 2 is currently installed on the NCAR Cray computers.

DOCUMENTS

No separate manual for SLATEC exists. The Consulting Office has copies of a locally-prepared document containing a brief description of all subprograms. A list of brief summaries of all the routine names and purposes, can be obtained by running a Cray job that includes the following command:

```
GETDOC,LIB=SLATEC,DOC=SUMMARY.
```

To list the online write-up for a SLATEC routine, say the routine named SGEFS, run a Cray job that includes the following command:

```
GETDOC,LIB=SLATEC,DOC=SGEFS,L=$OUT.
```
SLATEC is available as an object library on the Cray.

To access the Cray object library, include SLATEC in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=SLATEC.

loads any needed routines from SLATEC after loading from the default libraries. Single precision and complex LINPACK and EISPACK 2.0 routines have been removed from the SLATEC object library to make sure that users get the benefits of the optimized versions of these routines on the Cray default library $SCILIB.

To access SLATEC FORTRAN source files from the Cray, use the GETSRC utility. For example, to place the source for SGEFS and SGEIR in the Cray local dataset SRC, include the following command in the Cray job:

GETSRC,LIB=SLATEC,FILE=(SGEFS:SGEIR),L=SRC.

SLATEC is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for SLATEC. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in SLATEC.
ORIGIN

STARPAC (The Standards Time Series and Regression Package) is a library of FORTRAN subroutines for statistical data analysis developed by the Statistical Engineering Division (SED) of the National Bureau of Standards (NBS), Boulder, Colorado. Earlier versions of this library were distributed by the SED under the name STATLIB [Tryon and Donaldson, 1978]. STARPAC incorporates many changes to STATLIB, including additional statistical techniques, improved algorithms, and enhanced portability. It was acquired by NCAR in January 1984.

The STARPAC source code is being released in segments. The first segment (and currently the only segment) contains routines for nonlinear least squares regression. These subroutines:

- Accept user-supplied derivatives when available, otherwise numerically approximate the derivatives;
- Provide utility procedures to verify the correctness of the user-supplied derivatives, or to select optimum step sizes for numerically approximating the derivative;
- Allow both weighted and unweighted analyses;
- Permit subsets of the model parameters to be treated as constants with their values held fixed at their input values, allowing the user to examine the results obtained by estimating subsets of the parameters of a general model without rewriting the model subroutine; and
- Provide three levels of user-control of the computations and results, extensive error handling facilities, comprehensive printed reports, and no size restrictions other than effective machine size.

Users should contact the consulting office to see if additional STAR PAC segments have been released.

DOCUMENTS

The documentation for STARPAC consists of two NBS Technical Notes, Introduction to STARPAC, NBS Technical Note 1068-1, and Nonlinear Least Squares Regression Using STARPAC, NBS Technical Note 1068-2. Reference copies of these manuals are retained in the Consulting Office Library (Room 9C). Users may obtain their own copies by contacting the Consulting Office.
To list all of the subroutine descriptions, including STAR PAC under the category “E2b. Nonlinear Least Squares Approximations”, run a Cray job that includes the following command:

GETDOC,LIB=DOCLIB,DOC=E2B.

To list the online summary of a STAR PAC routine, say the routine named NLS, run a Cray job that includes the following command:

GETDOC,LIB=STAR PAC,DOC=NLS.

**ACCESS**

STAR PAC is available as an object library on the Cray.

To access the Cray object library, include STAR PAC in the list of parameter values for the LIB keyword on the LDR statement. For example

LDR,LIB=STAR PAC.

loads any needed routines from STAR PAC after loading from the default libraries.

To access STAR PAC FORTRAN source files from the Cray, use the GETSRC utility. For example, to list the source for NLS and NLSC, include the following command in the Cray job:

GETSRC,LIB=STAR PAC,FILE=(NLS:NLSC),L=$OUT.

**SUPPORT**

STAR PAC is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for STAR PAC. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray in STAR PAC.
STATLIB

ORIGIN  STATLIB is a library of FORTRAN subroutines for statistical analysis of experimental data, acquired from the Statistical Engineering Division at Boulder of the National Bureau of Standards. Its availability was announced at NCAR in March 1979. STATLIB is designed to be an easy-to-use and powerful tool for data analysis. Primary emphasis has been placed on the interpretation of statistical analyses. For this reason comprehensive automatic printout of auxiliary statistical information, often in graphical form, is provided to augment the basic statistical computations.

A fundamental STATLIB philosophy is to provide two or more subroutines for each method of analysis. A "simple" subroutine using extensive preset parameters minimizes the complexity of the call statement and gives a comprehensive printout of the results. Other subroutines provide complete control of the computations, suppression of all or part of the printout, and provide storage of computed results for further use.

At present STATLIB contains subroutines covering the following data analysis techniques:

- Plotting Data
- Analysis of a Single Random Sample
- The comparison of Two or More Random Samples
- Correlation Analysis
- Linear Least Squares Analysis
- Nonlinear Least Squares Analysis
- Time Series Analysis
- Random Data Generation

STAR PAC routines for nonlinear least squares supersede STATLIB nonlinear least squares routines and should be used instead. STATLIB will be removed from the system soon after all the STAR PAC routines corresponding to STATLIB routines are available (expected late 1986). Check with the Consulting Office for updated information.

DOCUMENTS  The Consulting Office has a copy of the users' manual for STATLIB

STATLIB -- A Library of FORTRAN Subroutines for Statistical Analysis of Experimental Data
by Peter V. Tryon and Janet R. Donaldson

which contains writeups for each routine in STATLIB. Users may obtain their own copies by contacting the Consulting Office. No
on-line writeups are available for STATLIB routines.

ACCESS

STATLIB is available as an object library on the Cray–1.

To access the Cray–1 object library, include STATLIB in the list of parameter values for the LIB keyword on the LDR statement. For example

\[ \text{LDR,LIB=STATLIB}. \]

would load any needed routines from STATLIB after loading from the default libraries.

The FORTRAN source code for STATLIB is in a file on the PSTORE - /TB/STATLIB/X205.FOR. Since it is stored as a single large file, individual subroutines in the file cannot be conveniently listed without first looking up their location within the file. A listing of the STATLIB source file is maintained in the Consulting Office.

SUPPORT

STATLIB is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any vendor-supplied updates. Documentation and usage statistics are maintained for STATLIB. The NCAR Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation. No attempts have been made to exploit the vectorization capabilities of the Cray–1 in STATLIB. Questions about the algorithms used or corrections to the routines should be directed to the Consulting Office before contacting the Statistical Engineering Division/Boulder of the National Bureau of Standards.
$SYSLIB is a library of input/output and system support routines maintained by Cray Research, Inc. Most of these routines are written in CAL (Cray Assembly Language) and they are only available on the Cray-1. $SYSLIB routines usually link directly to the Cray Operating System (COS) through an exchange jump. Many of these routines are not accessible from Fortran programs, but are called by $FTLIB routines for specific tasks. In general, $SYSLIB exists to serve as a link between the general purpose $FTLIB routines and the details of COS. $SYSLIB routines depend on specific COS features. Routine descriptions in this Catalog are for version 1.14 of $SYSLIB.

$SYSLIB routines are documented in

Cray-1 Library Reference Manual
CRI Publication SR-0014

The comment-block write-up at the beginning of each file may be listed using the GETDOC utility on the Cray. For example, to list the on-line write-up for ACCESS, include the following command in a Cray job:

GETDOC,LIB=$SYSLIB,DOC=ACCESS.

The name used for a $SYSLIB routine in the NCAR Library Catalog is the name of the source UPDATE-deck name. This is not always the same as the name of the user-callable subroutine. For example, GTDSP is the UPDATE-deck name for source for the entry GETDSP. A list of UPDATE-deck names and corresponding entries may be found in Chapter 2 of the Cray Library Reference Manual.

Routines in $SYSLIB are accessed automatically by the Cray loader. Some are explicitly callable from a FORTRAN program (e.g. OPEN, DISPOSE). Others are callable only from CAL (e.g. $ADW, $PRCW). Most of these latter subroutines are intended for system support only and are not otherwise useful. These may be accessed and listed by using the GETSRC command. For example, to place the source for ACCESS on Cray local dataset SRC, include the following command in a Cray-1 job:

GETSRC,LIB=$SYSLIB,FILE=ACCESS,L=SRC.
$SYSLIB is a fully supported Cray product. Problems with $SYSLIB routines should be brought to the attention of the Consulting Office. A consultant will verify the problem and report it to the Cray site analyst.
Most of the utilities described in this catalog are provided in the Cray Operating System. Many of the remaining, non-Cray utilities have been locally developed. Others, e.g. the VOS Software Tools, have been acquired externally.

The VOS Tools are a collection of utilities which are descendents of those developed in the book “Software Tools” by Brian Kernighan and P. J. Plauger and enhanced by members of the Software Tools Users Group. The current version of these utilities is described in an article, “A Virtual Operating System” by Dennis E. Hall, Deborah K. Scherrer and Joseph S. Sventek (CACM, Volume 23, Number 9, September 1980, page 495). Due to this heritage, the package of tools and associated utilities is sometimes referred to as the “Software Tools” and other times as the “Virtual Operating System (VOS) Tools”.

The Cray utilities are described in the Cray Reference Manuals (COS, CFT, and Library); some of the locally-developed utilities or externally acquired utilities are described in hardcopy available from the NCAR Consulting Office.

The Consulting Office has a copy of “The Software Tools Programmers Manual”, which contains writeups for all the primitive routines, utility routines and tools programs that make up the VOS Tools package, is available in the SCD consultation office.

On the NCAR IBM 4341, the CMS version of the tools programs, library routines and documentation are on the NCARLIBS machine TOOLS minidisk. This disk can be accessed with the command

```
LINKLIB TOOLS
```

Having accessed this disk, the documentation for any of the VOS-TOOLS routines can be obtained using the CMS HELP facility. For example, the documentation for the text formatter ROFF would be obtained with the command:

```
HELP ROFF
```

or
```
HELP TOOLS ROFF
```

No explicit access statement is necessary to use any of the Cray utilities or locally-developed utilities. Cray access for usage of the VOS Software Tools is achieved with Cray JCL statement

```
TOOLS.
```

At this time, the GETSRC utility has not been developed to include source access for any of the utilities, and there are no plans to do so.
Users who wish to obtain source for any of the utilities should contact Richard Valent of the NCAR Software and Libraries Group.

**SUPPORT**

CRAY Research, Inc. supports its utilities in conjunction with the NCAR CRAY Analysts and NCAR Systems Group. The NCAR Software and Libraries Group supports its externally acquired software at the level of applying any author- or vendor-supplied updates. Its locally developed software is supported at the highest level.

The VOS Tools package is supported at a level appropriate for high-quality externally acquired software. NCAR takes no responsibility for correcting errors, but will apply any externally supplied updates. The Software and Libraries Group does not guarantee that the software will survive the acquisition of new computers or operating systems, although, by their very nature, there is a high probability that the VOS tools and associated library will already have been implemented on any newly acquired systems. No local expertise is guaranteed, other than familiarity of consultants with the documentation.