



# IMSL® C Numerical Library Function Catalog

## Version 2016.1

At the heart of the IMSL C Numerical Library is a comprehensive set of pre-built mathematical and statistical analysis functions that developers can embed directly into their applications. Available for a wide range of computing platforms, the robust, scalable, portable and high performing IMSL analytics allow developers to focus on their domain of expertise and reduce development time.

## COST-EFFECTIVENESS AND VALUE

The IMSL C Numerical Library significantly shortens application time to market and promotes standardization. Descriptive function names and variable argument lists have been implemented to simplify calling sequences. Using the IMSL C Library reduces costs associated with the design, development, documentation, testing and maintenance of applications. Robust, scalable, portable, and high performing analytics with IMSL forms the foundation for inherently reliable applications.

## A RICH SET OF PREDICTIVE ANALYTICS FUNCTIONS

The library includes a comprehensive set of functions for machine learning, data mining, prediction, and classification, including:

- Time series models such as ARIMA, GARCH, and VARMA vector auto-regression.
- Predictive models such as decision trees, random forest, stochastic gradient boosting, support vector machines, neural networks, linear and multinomial logistic regression, and Bayes classification.
- Data mining algorithms such as K-means and hierarchical clustering, Apriori market basket analysis and much more.

The IMSL C Library also includes functions for analyzing streaming data and working with big or distributed data.

## A SUITE OF OPTIMIZATION FUNCTIONS

The library includes a suite of optimization methods for different problems including multivariate, derivative free, constrained or unconstrained, dense or sparse, linear or general objective functions. The library provides numerous functions for solving linear systems and systems of differential equations and offers broad coverage for mathematical analysis, including transforms, convolutions, splines, quadrature, and more.

## EMBEDDABILITY

Development is made easier because library code readily embeds into application code, with no additional infrastructure such as app/management consoles, servers, or programming environments needed.

Wrappers complicate development by requiring the developer to access external compilers and pass arrays or user-defined data types to ensure compatibility between the different languages. The IMSL C Library allows developers to write, build, compile and debug code in a single environment, and to easily embed analytic functions in applications and databases.

## RELIABILITY

100% pure C code maximizes robustness. It allows faster and easier debugging, through superior error handling that not only conveys the error condition, but also suggests corrective action if appropriate. The result is reduced application support cost due to the minimization of user error and data issues.

The IMSL C Library has been rigorously tested by Rogue Wave and seasoned by all industry verticals for over 40 years. You can expect consistent results across all supported platforms and languages, which makes platform migration and upgrade easy and efficient.

## HIGH PERFORMANCE

The IMSL C Library integrates third-party, high-performance vendor BLAS libraries. For many linear algebra functions, work can be offloaded to the vendor library for enhanced performance. The calling sequences for the IMSL functions are unchanged, so there is no learning curve and users can be productive immediately.

The library is also designed to take advantage of symmetric multiprocessor (SMP) systems, both multi-CPU and multi-core. Many algorithms leverage OpenMP directives on supported environments to distribute calculations across available resources. In

areas such as linear algebra and Fast Fourier Transforms, third-party high-performance vendor libraries leverage SMP capabilities on a variety of systems.

### SCALABILITY

The IMSL C Library supports scalability through:

- An enhanced ability to analyze time-sequenced data, or streaming, real-time data that isn't stored.
- Improved algorithms that can analyze data sets too large to fit into memory or that exist on separate nodes.
- Memory management that ensures applications will not crash when they encounter a low memory condition.

### USER FRIENDLY NOMENCLATURE

The IMSL C Library uses descriptive, explanatory function names for intuitive programming that:

- Are easy to identify and use, and prevent conflicts with other software.
- Provide a common root name for numerical functions that offer the choice of multiple precisions.

### PROGRAMMING INTERFACE FLEXIBILITY

The IMSL C Library takes full advantage of the intrinsic characteristics and desirable features of the C language.

The functions support variable-length argument lists, where the concise set of required arguments contains only information necessary for usage. Optional arguments provide added functionality and power to each function. Memory allocation can be handled by the library or by the developer. Finally, user-defined functions are implemented with interfaces C developers will find natural.

### THREAD SAFETY

The IMSL C Library is thread-safe. Thread safety allows the C Library to be used in multi-threaded applications where performance benefits can be realized through concurrent and/or parallel execution.

### COMPREHENSIVE DOCUMENTATION

Documentation for the IMSL C Numerical Library is comprehensive, clearly written and standardized. The documentation, in multiple formats:

- Provides organized, easy-to-find information.
- Documents, explains, and provides references for algorithms.
- Gives at least one example of each function's usage, with sample input and result.

### UNMATCHED PRODUCT SUPPORT

Behind every Rogue Wave license is a team of professionals ready to provide expert answers to questions about the IMSL Numerical Libraries.

Product support:

- Gives users direct access to Rogue Wave's resident staff of expert product support specialists.
- Provides prompt, two-way communication with solutions to a user's programming needs.
- Includes product maintenance updates.

### PROFESSIONAL SERVICES

Rogue Wave offers expert consulting services for algorithm development as well as complete application development. Please contact Rogue Wave to learn more about its extensive experience in developing custom algorithms, building algorithms on scalable platforms, and full applications development.

### ACCESS FROM PYTHON

The IMSL C Library functions is also available within Python for rapid prototyping, and a collection of Python wrappers to the algorithms in the IMSL C Numerical Library are available on request. By using the same IMSL algorithms in the prototype as in the production code, developers can deliver accurate and consistent results in the production application sooner rather than later. For more information, please contact [info@roguewave.com](mailto:info@roguewave.com).

## Mathematical Functionality Overview

The IMSL C Numerical Library is a collection of the most commonly required numerical functions, tailored for a C programmer's needs. The mathematical functionality is organized into 12 sections. These capabilities range from solving systems of linear equations to optimization.

**Linear Systems** - including real and complex, full and sparse matrices, linear least squares, matrix decompositions, generalized inverses and vector-matrix operations. The least squares solvers may include non-negative and general linear constraints. Matrix decompositions now include non-negative, low-rank factorizations.

**Eigensystems Analysis** - including eigenvalues and eigenvectors of complex, real symmetric and complex Hermitian matrices.

**Interpolation and Approximation** - including constrained curve-fitting splines, cubic splines, least-squares approximation and smoothing, and scattered data interpolation.

**Integration and Differentiation** - including univariate, multivariate, Gauss quadrature and quasi-Monte Carlo.

**Differential Equations** - using Adams-Gear and Runge-Kutta methods for stiff and non-stiff ordinary differential equations, with support for partial differential equations, including the Feynman-Kac solver. Also included are second-order ODE solvers, constrained DAE solvers, and a method of lines PDE solver.

**Transforms** - including real and complex, one- and two-dimensional Fast Fourier Transforms, as well as convolutions, correlations and Laplace transforms.

**Nonlinear Equations** - including zeros and root finding of polynomials, zeros of a function and root of a system of equations.

**Optimization** - including unconstrained linearly and nonlinearly constrained minimizations and linear and quadratic programming interior point algorithms.

**Special Functions** - including error and gamma functions, real order complex valued Bessel functions and statistical functions.

**Financial Functions** - including functions for Bond and cash-flow analysis.

**Random Number Generation** - including a generator for multivariate normal distributions and pseudorandom numbers from several distributions, including gamma, Poisson and beta. Also, support for low discrepancy series using a generalized Faure sequence.

**Utilities** - including CPU time used, machine, mathematical, physical constants, retrieval of machine constants and customizable error-handling.

## Statistical Functionality Overview

The statistical functionality is organized into thirteen sections. These capabilities range from analysis of variance to random number generation.

**Basic Statistics** - including univariate summary statistics, frequency tables, and rank and order statistics.

**Regression** - including linear (multivariate), polynomial, and nonlinear regression models as well as robust alternatives such as Lpnorm and Partial Least Squares. This section also contains stepwise and all best variable selection methods.

**Correlation and Covariance** - including sample variance-covariance, partial correlation and covariances, pooled variance-covariance and robust estimates of a covariance matrix and mean factor.

**Analysis of Variance and Designed Experiments** - analysis of standard factorial experiments, randomized complete block designs, Latin-square, lattice, split-plot and strip-plot and related experiments; analysis of hierarchical data, computation of false discovery rates and standard tests for multiple comparisons and homogeneity of variance.

**Categorical and Discrete Data Analysis** - including chi-squared analysis of a two-way contingency table, exact probabilities in a two-way contingency table, logistic regression for binomial or multinomial responses, and the analysis of categorical data using general linear models.

**Nonparametric Statistics** - including sign tests, Wilcoxon sum tests and Cochran Q test for related observations.

**Tests of Goodness-of-Fit** - including the chi-squared goodness-of-fit test, the Kolmogorov/Smirnov one- and two-sample tests for continuous distributions, the Shapiro-Wilk, Lilliefors, chi-squared, Anderson-Darling, and Cramer-Von Mises tests for normality, Mardia's test for multivariate normality, and the runs, pairs-serial, d2, and triplets tests for randomness.

**Time Series Analysis and Forecasting** - analysis and forecasting of time series using a nonseasonal ARMA model, ARIMA with regression, Holt-Winters exponential smoothing, GARCH, Kalman filtering; various fitting and diagnostic utilities including portmanteau lack of fit test and difference of a seasonal or nonseasonal time series.

**Multivariate Analysis** - including principal component analysis, discriminant analysis, K-means and hierarchical cluster analysis, and factor analysis. Methods of factor analysis include principal factor, image analysis, unweighted least squares, generalized least squares, maximum likelihood, and various factor rotations.

**Survival Analysis** - including analysis of data using the Cox linear survival model, Kaplan-Meier survival estimates, actuarial survival tables, and non-parametric survival estimates.

**Probability Distribution Functions and Inverses** - including the cumulative distribution function (CDF), inverse CDF, and probability density function (PDF) for many common discrete and continuous distributions, as well as multivariate normal, non-central F, Chi-square, Beta, Students t, and others. This section also includes parameter estimation by maximum likelihood.

**Random Number Generation** - including generators for many univariate discrete and continuous distributions, as well as multivariate Normal, multinomial, Gaussian or Student's t copula, an ARMA or nonhomogeneous Poisson process, order statistics, permutations, and more. This section also allows a choice of pseudorandom number generators, including the Mersenne Twister.

**Data Mining** - including decision trees, vector auto-regression, Apriori analysis, cluster analysis, Kohonen self-organizing maps, support vector machine, ensemble models, genetic algorithms, PCA, factor analysis, feed-forward neural networks, neural network data pre- and post-processing algorithms, and much more.

## IMSL® Libraries are also available for Fortran, Java, and C# for .NET

### IMSL® FORTRAN NUMERICAL LIBRARY

The IMSL® Fortran Numerical Library is the gold standard mathematical and statistical code library for Fortran programmers developing high performance computing applications. The IMSL Fortran Library contains highly accurate and reliable Fortran algorithms with full coverage of mathematics and statistics and complete backward compatibility.

The IMSL Fortran Library is a comprehensive library of mathematical and statistical algorithms available in one cohesive package. It combines the powerful and flexible interface features of the Fortran language with the performance benefits of both distributed memory and shared memory multiprocessing architectures.

### JMSL™ NUMERICAL LIBRARY FOR JAVA APPLICATIONS

The JMSL Numerical Library for Java applications is the broadest collection of mathematical, statistical, financial, data mining and charting classes available in 100% Java. It is the only Java programming solution that combines integrated charting with the reliable mathematical and statistical functionality of the industry-leading IMSL Numerical Library algorithms. This blend of advanced numerical analysis and visualization on the Java platform allows organizations to gain insight into valuable data and share analysis results across the enterprise quickly. The JMSL Library continues to be the leader, providing robust and portable data analysis and visualization technology for the Java platform, and a fast, scalable framework for tailored analytical applications.

### IMSL C# NUMERICAL LIBRARY FOR MICROSOFT® .NET APPLICATIONS

The IMSL C# Numerical Library for Microsoft® .NET Applications is a numerical analysis library written in 100% C#, providing broad coverage of advanced mathematics and statistics for the .NET Framework. This offers .NET developers seamless accessibility to analytics capabilities in the most integrated language for the .NET environment with the highest degree of programming productivity and ease of use with Visual Studio®.

The IMSL C# Library is the only numerical library to offer industry standard numerical analysis and charting for .NET languages. This Library provides unprecedented analytic capabilities and the most comprehensive and accessible mathematical, statistical and finance algorithms for .NET languages. With the IMSL C# Library, IMSL has brought all of the benefits inherent in the .NET Framework to a new level by adding robust analytics to its broad set of capabilities. Because the C# Library is written in C#, it is easily integrated into any .NET language, such as Visual Basic .NET, F# and IronPython among others.



# IMSL C MATH LIBRARY

## CHAPTER 1: LINEAR SYSTEMS

### LINEAR EQUATIONS WITH FULL MATRICES

FUNCTION	DESCRIPTION
<a href="#">lin_sol_gen</a>	Solves a real general system of linear equations $Ax = b$ .
<a href="#">lin_sol_gen (complex)</a>	Solves a complex general system of linear equations $Ax = b$ .
<a href="#">lin_sol_posdef</a>	Solves a real symmetric positive definite system of linear equations $Ax = b$ .
<a href="#">lin_sol_posdef (complex)</a>	Solves a complex Hermitian positive definite system of linear equations $Ax = b$ .

### LINEAR EQUATIONS WITH BAND MATRICES

FUNCTION	DESCRIPTION
<a href="#">lin_sol_gen_band</a>	Solves a real general band system of linear equations $Ax = b$ .
<a href="#">lin_sol_gen_band (complex)</a>	Solves a complex general band system of linear equations $Ax = b$ .
<a href="#">lin_sol_posdef_band</a>	Solves a real symmetric positive definite system of linear equations $Ax = b$ in band symmetric storage mode.
<a href="#">lin_sol_posdef_band (complex)</a>	Solves a complex Hermitian positive definite system of linear equations $Ax = b$ in band symmetric storage mode.

LINEAR EQUATIONS WITH GENERAL SPARSE MATRICES	
FUNCTION	DESCRIPTION
<b>lin_sol_gen_coordinate</b>	Solves a sparse system of linear equations $Ax = b$ .
<b>lin_sol_gen_coordinate (complex)</b>	Solves a system of linear equations $Ax = b$ , with sparse complex coefficient matrix $A$ .
<b>superlu</b>	Computes the $LU$ factorization of a general sparse matrix by a column method and solves the real sparse linear system of equations $Ax = b$ .
<b>superlu (complex)</b>	Computes the $LU$ factorization of a general complex sparse matrix by a column method and solves the complex sparse linear system of equations $Ax = b$ .
<b>superlu_smp</b>	Computes the $LU$ factorization of a general sparse matrix by a left-looking column method using OpenMP parallelism, and solves the real sparse linear system of equations $Ax = b$ .
<b>superlu_smp (complex)</b>	Computes the $LU$ factorization of a general complex sparse matrix by a left-looking column method using OpenMP parallelism and solves the complex sparse linear system of equations $Ax = b$ .
<b>lin_sol_posdef_coordinate</b>	Solves a sparse real symmetric positive definite system of linear equations $Ax = b$ .
<b>lin_sol_gen_posdef_coordinate (complex)</b>	Solves a sparse Hermitian positive definite system of linear equations $Ax = b$ .
<b>sparse_cholesky_smp</b>	Computes the Cholesky factorization of a sparse real symmetric positive definite matrix $A$ by an OpenMP parallelized supernodal algorithm and solves the sparse real positive definite system of linear equations $Ax = b$ .
<b>sparse_cholesky_smp (complex)</b>	Computes the Cholesky factorization of a sparse (complex) Hermitian positive definite matrix $A$ by an OpenMP parallelized supernodal algorithm and solves the sparse Hermitian positive definite system of linear equations $Ax = b$ .
<b>lin_sol_gen_min_residual</b>	Solves a linear system $Ax = b$ using the restarted generalized minimum residual (GMRES) method.
<b>lin_sol_def_cg</b>	Solves a real symmetric definite linear system using a conjugate gradient method.

ITERATIVE METHODS	
FUNCTION	DESCRIPTION
<a href="#">lin_sol_gen_min_residual</a>	Solves a linear system $Ax = b$ using the restarted generalized minimum residual (GMRES) method.
<a href="#">lin_sol_def_cg</a>	Solves a real symmetric definite linear system using a conjugate gradient method.

LINEAR LEAST-SQUARES WITH FULL MATRICES	
FUNCTION	DESCRIPTION
<a href="#">lin_least_squares_gen</a>	Solves a linear least-squares problem $Ax = b$ .
<a href="#">nonneg_least_squares</a>	Computes the non-negative least squares (NNLS) solution of an $m \times n$ real linear least squares system, $Ax \cong b, x \geq 0$ .
<a href="#">lin_lsq_lin_constraints</a>	Solves a linear least-squares problem with linear constraints.
<a href="#">nonneg_matrix_factorization</a>	Given an $m \times n$ real matrix $A \geq 0$ and an integer $k \leq \min(m, n)$ , computes a factorization $A \cong FG$ .
<a href="#">lin_svd_gen</a>	Computes the SVD, $A = USV^T$ , of a real rectangular matrix $A$ .
<a href="#">lin_svd_gen (complex)</a>	Computes the SVD, $A = USV^T$ , of a complex rectangular matrix $A$ .
<a href="#">lin_sol_nonnegdef</a>	Solves a real symmetric nonnegative definite system of linear equations $Ax = b$ .

## CHAPTER 2: EIGENSYSTEM ANALYSIS

### LINEAR EIGENSYSTEM PROBLEMS

FUNCTION	DESCRIPTION
<b>eig_gen</b>	Computes the eigenexpansion of a real matrix $A$ .
<b>eig_gen (complex)</b>	Computes the eigenexpansion of a complex matrix. $A$ .
<b>eig_sym</b>	Computes the eigenexpansion of a real symmetric matrix $A$ .
<b>eig_herm (complex)</b>	Computes the eigenexpansion of a complex Hermitian matrix $A$ .

### GENERALIZED EIGENSYSTEM PROBLEMS

FUNCTION	DESCRIPTION
<b>eig_symgen</b>	Computes the generalized eigenexpansion of a system $Ax = \lambda Bx$ . $A$ and $B$ are real and symmetric. $B$ is positive definite.
<b>geneig</b>	Computes the generalized eigenexpansion of a system $Ax = \lambda Bx$ , with $A$ and $B$ real.
<b>geneig (complex)</b>	Computes the generalized eigenexpansion of a system $Ax = \lambda Bx$ , with $A$ and $B$ complex.

## CHAPTER 3: INTERPOLATION AND APPROXIMATION

### CUBIC SPLINE INTERPOLATION

FUNCTION	DESCRIPTION
<code>cub_spline_interp_e_cnd</code>	Computes a cubic spline interpolant, specifying various endpoint conditions.
<code>cub_spline_interp_shape</code>	Computes a shape-preserving cubic spline.
<code>cub_spline_tcb</code>	Computes a tension-continuity-bias (TCB) cubic spline interpolant. This is also called a Kochanek- Bartels spline and is a generalization of the Catmull-Rom spline.

### CUBIC SPLINE EVALUATION AND INTEGRATION

FUNCTION	DESCRIPTION
<code>cub_spline_value</code>	Computes the value of a cubic spline or the value of one of its derivatives.
<code>cub_spline_integral</code>	Computes the integral of a cubic spline.

### SPLINE INTERPOLATION

FUNCTION	DESCRIPTION
<code>spline_interp</code>	Computes a spline interpolant.
<code>spline_knots</code>	Computes the knots for a spline interpolant.
<code>spline_2d_interp</code>	Computes a two-dimensional, tensor-product spline interpolant from two-dimensional, tensor-product data.

### SPLINE EVALUATION AND INTEGRATION

FUNCTION	DESCRIPTION
<a href="#">spline_value</a>	Computes the value of a spline or the value of one of its derivatives.
<a href="#">spline_integral</a>	Computes the integral of a spline.
<a href="#">spline_2d_value</a>	Computes the value of a tensor-product spline or the value of one of its partial derivatives.
<a href="#">spline_2d_integral</a>	Evaluates the integral of a tensor-product spline on a rectangular domain.

### MULTI-DIMENSIONAL INTERPOLATION

FUNCTION	DESCRIPTION
<a href="#">spline_nd_interp</a>	Performs multidimensional interpolation and differentiation for up to 7 dimensions.

### LEAST-SQUARES APPROXIMATION AND SMOOTHING

FUNCTION	DESCRIPTION
<a href="#">user_fcn_least_squares</a>	Computes a least-squares fit using user-supplied functions.
<a href="#">spline_least_squares</a>	Computes a least-squares spline approximation.
<a href="#">spline_2d_least_squares</a>	Computes a two-dimensional, tensor-product spline approximant using least squares.

## LEAST-SQUARES APPROXIMATION AND SMOOTHING (CONTINUED)

FUNCTION	DESCRIPTION
<a href="#">cub_spline_smooth</a>	Computes a smooth cubic spline approximation to noisy data by using cross-validation to estimate the smoothing parameter or by directly choosing the smoothing parameter.
<a href="#">spline_lsq_constrained</a>	Computes a least-squares constrained spline approximation.
<a href="#">smooth_1d_data</a>	Smooth one-dimensional data by error detection.

## SCATTERED DATA INTERPOLATION

FUNCTION	DESCRIPTION
<a href="#">scattered_2d_interp</a>	Computes a smooth bivariate interpolant to scattered data that is locally a quintic polynomial in two variables.

## SCATTERED DATA LEAST SQUARES

FUNCTION	DESCRIPTION
<a href="#">radial_scattered_fit</a>	Computes an approximation to scattered data in $R^n$ for $n \geq 1$ using radial-basis functions.
<a href="#">radial_evaluate</a>	Evaluates a radial basis fit.

## CHAPTER 4: QUADRATURE

UNIVARIATE QUADRATURE	
FUNCTION	DESCRIPTION
<code>int_fcn_sing</code>	Integrates a function, which may have endpoint singularities, using a globally adaptive scheme based on Gauss–Kronrod rules.
<code>int_fcn_sing_1d</code>	Integrates a function with a possible internal or endpoint singularity.
<code>int_fcn</code>	Integrates a function using a globally adaptive scheme based on Gauss–Kronrod rules.
<code>int_fcn_sing_pts</code>	Integrates a function with singularity points given.
<code>int_fcn_alg_log</code>	Integrates a function with algebraic-logarithmic singularities.
<code>int_fcn_inf</code>	Integrates a function over an infinite or semi-infinite interval.
<code>int_fcn_trig</code>	Integrates a function containing a sine or a cosine factor.
<code>int_fcn_fourier</code>	Computes a Fourier sine or cosine transform.
<code>int_fcn_cauchy</code>	Computes integrals of the form $\int_a^b \frac{f(x)}{x-c} dx$ in the Cauchy principal value sense.
<code>int_fcn_smooth</code>	Integrates a smooth function using a nonadaptive rule.



## MULTIDIMENSIONAL QUADRATURE

FUNCTION	DESCRIPTION
<a href="#">int_fcn_2d</a>	Computes a two-dimensional iterated integral.
<a href="#">int_fcn_sing_2d</a>	Integrates a function of two variables with a possible internal or endpoint singularity.
<a href="#">int_fcn_sing_3d</a>	Integrates a function of three variables with a possible internal or endpoint singularity.
<a href="#">int_fcn_hyper_rect</a>	Integrate a function on a hyper-rectangle, $\int_{a_0}^{b_0} \dots \int_{a_{n-1}}^{b_{n-1}} f(x_0, \dots, x_{n-1}) dx_{n-1} \dots dx_0$
<a href="#">int_fcn_qmc</a>	Integrates a function over a hyper-rectangle using a quasi-Monte Carlo method.

## GAUSS RULES AND THREE-TERM RECURRENCES

FUNCTION	DESCRIPTION
<a href="#">gauss_quad_rule</a>	Computes a Gauss, Gauss–Radau, or Gauss–Lobatto quadrature rule with various classical weight functions.

## DIFFERENTIATION

FUNCTION	DESCRIPTION
<a href="#">fcn_derivative</a>	Computes the first, second or third derivative of a user-supplied function.

## CHAPTER 5: DIFFERENTIAL EQUATIONS

### FIRST-ORDER ORDINARY DIFFERENTIAL EQUATIONS

#### SOLUTION OF THE INITIAL VALUE PROBLEM FOR ODES

FUNCTION	DESCRIPTION
<a href="#">ode_runge_kutta</a>	Solves an initial-value problem for ordinary differential equations using the Runge–Kutta–Verner fifth-order and sixth-order method.

#### SOLUTION OF THE BOUNDARY VALUE PROBLEM FOR ODES

FUNCTION	DESCRIPTION
<a href="#">bvp_finite_difference</a>	Solves a (parameterized) system of differential equations with boundary conditions at two points, using a variable order, variable step size finite difference method with deferred corrections.

#### SOLUTION OF DIFFERENTIAL-ALGEBRAIC SYSTEMS

FUNCTION	DESCRIPTION
<a href="#">differential_algebraic_eqs</a>	Solves a first order differential-algebraic system of equations, $g(t, y, y') = 0$ , with optional additional constraints and user-defined linear system solver.

### FIRST-AND-SECOND ORDER ORDINARY DIFFERENTIAL EQUATIONS

#### SOLUTION OF THE INITIAL VALUE PROBLEM FOR ODES

FUNCTION	DESCRIPTION
<a href="#">ode_adams_krogh</a>	Solves an initial-value problem for a system of ordinary differential equations of order one or two using a variable order Adams method.

## PARTIAL DIFFERENTIAL EQUATIONS

## SOLUTION OF SYSTEMS OF PDES IN ONE DIMENSION

FUNCTION	DESCRIPTION
<a href="#">pde_1d_mg</a>	Solves a system of one-dimensional time-dependent partial differential equations using a moving-grid interface.
<a href="#">modified_method_of_lines</a>	Solves a system of partial differential equations of the form $u_t = f(x, t, u, u_x, u_{xx})$ using the method of lines.
<a href="#">feynman_kac</a>	Solves a generalized Feynman-Kac equation on a finite interval using Hermite quintic splines.
<a href="#">feynman_kac_evaluate</a>	Computes the value of a Hermite quintic spline or the value of one of its derivatives.

## SOLUTION OF A PDE IN TWO DIMENSIONS

FUNCTION	DESCRIPTION
<a href="#">fast_poisson_2d</a>	Solves Poisson's or Helmholtz's equation on a two-dimensional rectangle using a fast Poisson solver based on the HODIE finite-difference scheme on a uniform mesh.

## CHAPTER 6: TRANSFORMS

### REAL TRIGONOMETRIC FFT

FUNCTION	DESCRIPTION
<code>fft_real</code>	Computes the discrete Fourier transform of a real sequence.
<code>fft_real_init</code>	Computes the parameters for <code>imsl_f_fft_real</code> .

### COMPLEX EXPONENTIAL FFT

FUNCTION	DESCRIPTION
<code>fft_complex</code>	Computes the complex discrete Fourier transform of a complex sequence.
<code>fft_complex_init</code>	Computes the parameters for <code>imsl_c_fft_complex</code> .

### REAL SINE AND COSINE FFTS

FUNCTION	DESCRIPTION
<code>fft_cosine</code>	Computes the discrete Fourier cosine transformation of an even sequence.
<code>fft_cosine_init</code>	Computes parameters needed by <code>imsl_f_fft_cosine</code> .
<code>fft_sine</code>	Computes the discrete Fourier sine transformation of an odd sequence.

## REAL SINE AND COSINE FFTS (CONTINUED)

FUNCTION	DESCRIPTION
<code>fft_sine_init</code>	Computes parameters needed by <code>imsl_f_fft_cosine</code> .

## TWO-DIMENSIONAL FFTS

FUNCTION	DESCRIPTION
<code>fft_2d_complex</code>	Computes the complex discrete two-dimensional Fourier transform of a complex two-dimensional array.

## CONVOLUTIONS AND CORRELATIONS

FUNCTION	DESCRIPTION
<code>convolution</code>	Computes the convolution of two real vectors.
<code>convolution (complex)</code>	Computes the convolution, and optionally, the correlation of two complex vectors.

## LAPLACE TRANSFORM

FUNCTION	DESCRIPTION
<code>inverse_laplace</code>	Computes the inverse Laplace transform of a complex function.

## CHAPTER 7: NONLINEAR EQUATIONS

### ZEROS OF A POLYNOMIAL

FUNCTION	DESCRIPTION
<b>zeros_poly</b>	Finds the zeros of a polynomial with real coefficients using the Jenkins-Traub three-stage algorithm.
<b>zeros_poly (complex)</b>	Finds the zeros of a polynomial with complex coefficients using the Jenkins-Traub three-stage algorithm.

### ZEROS OF A FUNCTION

FUNCTION	DESCRIPTION
<b>zero_univariate</b>	Finds a zero of a real univariate function.
<b>zero_function</b>	Finds the real zeros of a real, continuous, univariate function.

### ROOT OF A SYSTEM OF EQUATIONS

FUNCTION	DESCRIPTION
<b>zeros_sys_eqn</b>	Solves a system of $n$ nonlinear equations $f(x) = 0$ using a modified Powell hybrid algorithm.

## CHAPTER 8: OPTIMIZATION

### UNCONSTRAINED MINIMIZATION

#### UNIVARIATE FUNCTION

FUNCTION	DESCRIPTION
<b>min_uncon</b>	Finds the minimum point of a smooth function $f(x)$ of a single variable using only function evaluations.
<b>min_uncon_deriv</b>	Finds the minimum point of a smooth function of a single variable using both function and first derivative evaluations.
<b>min_uncon_golden</b>	Finds the minimum point of a nonsmooth function of a single variable using the golden section search method.

#### MULTIVARIATE FUNCTION

FUNCTION	DESCRIPTION
<b>min_uncon_multivar</b>	Minimizes a function $f(x)$ of $n$ variables using a quasi-Newton method.
<b>min_uncon_polytope</b>	Minimizes a function of $n$ variables using a direct search polytope algorithm.

#### NONLINEAR LEAST SQUARES

FUNCTION	DESCRIPTION
<b>nonlin_least_squares</b>	Solves a nonlinear least-squares problem using a modified Levenberg-Marquardt algorithm.

LINEARLY CONSTRAINED MINIMIZATION	
FUNCTION	DESCRIPTION
<b>read_mps</b>	Reads an MPS file containing a linear programming problem or a quadratic programming problem.
<b>linear_programming</b>	Solves a linear programming problem.
<b>lin_prog</b>	Solves a linear programming problem using the revised simplex algorithm.
<b>quadratic_prog</b>	Solves a quadratic programming problem subject to linear equality or inequality constraints.
<b>sparse_lin_prog</b>	Solves a sparse linear programming problem by an infeasible primal-dual interior-point method.
<b>sparse_quadratic_prog</b>	Solves a sparse convex quadratic programming problem by an infeasible primal-dual interior-point method.
<b>min_con_gen_lin</b>	Minimizes a general objective function subject to linear equality/inequality constraints.
<b>bounded_least_squares</b>	Solves a nonlinear least-squares problem subject to bounds on the variables using a modified Levenberg-Marquardt algorithm.
<b>transport</b>	Solves a transportation problem.
<b>min_con_lin_trust_region</b>	Minimizes a function of n variables subject to linear constraints using a derivative-free, interpolation-based trust-region method.
<b>min_con_polytope</b>	Minimizes a function of n variables subject to bounds on the variables using a direct search complex algorithm.



## NONLINEARLY CONSTRAINED MINIMIZATION

FUNCTION	DESCRIPTION
<b>constrained_nlp</b>	Solves a general nonlinear programming problem using a sequential equality constrained quadratic programming method.

## SERVICE ROUTINES

FUNCTION	DESCRIPTION
<b>jacobian</b>	Approximates the Jacobian of $m$ functions in $n$ unknowns using divided differences.

## CHAPTER 9: SPECIAL FUNCTIONS

### ERROR AND GAMMA FUNCTIONS

#### ERROR FUNCTIONS

FUNCTION	DESCRIPTION
<b>erf</b>	Evaluates the real error function $\text{erf}(x)$ .
<b>erfc</b>	Evaluates the real complementary error function $\text{erfc}(x)$ .
<b>erf_inverse</b>	Evaluates the real inverse error function $\text{erf}^{-1}(x)$ .
<b>erfce</b>	Evaluates the exponentially scaled complementary error function.
<b>erfe</b>	Evaluates a scaled function related to $\text{erfc}(z)$ .
<b>erfc_inverse</b>	Evaluates the real inverse complementary error function $\text{erfc}^{-1}(x)$ .
<b>beta</b>	Evaluates the real beta function $\beta(x, y)$ .
<b>log_beta</b>	Evaluates the logarithm of the real beta function $\ln \beta(x, y)$ .
<b>beta_incomplete</b>	Evaluates the real incomplete beta function $I_x = \beta_x(a, b) / \beta(a, b)$ .

#### GAMMA FUNCTIONS

FUNCTION	DESCRIPTION
<b>gamma</b>	Evaluates the real gamma function $\Gamma(x)$ .
<b>log_gamma</b>	Evaluates the logarithm of the absolute value of the gamma function $\log  \Gamma(x) $ .

GAMMA FUNCTIONS	
<b>gamma_incomplete</b>	Evaluates the incomplete gamma function $\gamma(a, x)$ .

PSI FUNCTIONS	
FUNCTION	DESCRIPTION
<b>psi</b>	Evaluates the derivative of the log gamma function.
<b>psi1</b>	Evaluates the second derivative of the log gamma function.

BESSEL FUNCTIONS	
FUNCTION	DESCRIPTION
<b>bessel_J0</b>	Evaluates the real Bessel function of the first kind of order zero $J_0(x)$ .
<b>bessel_J1</b>	Evaluates the real Bessel function of the first kind of order one $J_1(x)$ .
<b>bessel_Jx</b>	Evaluates a sequence of Bessel functions of the first kind with real order and complex arguments.
<b>bessel_Y0</b>	Evaluates the real Bessel function of the second kind of order zero $Y_0(x)$ .
<b>bessel_Y1</b>	Evaluates the real Bessel function of the second kind of order one $Y_1(x)$ .
<b>bessel_Yx</b>	Evaluates a sequence of Bessel functions of the second kind with real order and complex arguments.
<b>bessel_I0</b>	Evaluates the real modified Bessel function of the first kind of order zero $I_0(x)$ .
<b>bessel_exp_I0</b>	Evaluates the exponentially scaled modified Bessel function of the first kind of order zero.
<b>bessel_I1</b>	Evaluates the real modified Bessel function of the first kind of order one $I_1(x)$ .

BESSEL FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<b>bessel_exp_I1</b>	Evaluates the exponentially scaled modified Bessel function of the first kind of order one.
<b>bessel_Ix</b>	Evaluates a sequence of modified Bessel functions of the first kind with real order and complex arguments.
<b>bessel_K0</b>	Evaluates the real modified Bessel function of the second kind of order zero $K_0(x)$ .
<b>bessel_exp_K0</b>	Evaluates the exponentially scaled modified Bessel function of the second kind of order zero.
<b>bessel_K1</b>	Evaluates the exponentially scaled modified Bessel function of the second kind of order one.
<b>bessel_exp_K1</b>	Evaluates the exponentially scaled modified Bessel function of the second kind of order one.
<b>bessel_Kx</b>	Evaluates a sequence of modified Bessel functions of the second kind with real order and complex arguments.

ELLIPTIC INTEGRALS	
FUNCTION	DESCRIPTION
<b>elliptic_integral_K</b>	Evaluates the complete elliptic integral of the kind $K(x)$ .
<b>elliptic_integral_E</b>	Evaluates the complete elliptic integral of the second kind $E(x)$ .
<b>elliptic_integral_RF</b>	Evaluates Carlson's elliptic integral of the first kind $R_F(x, y, z)$ .
<b>elliptic_integral_RD</b>	Evaluates Carlson's elliptic integral of the second kind $R_D(x, y, z)$ .
<b>elliptic_integral_RJ</b>	Evaluates Carlson's elliptic integral of the third kind $R_J(x, y, z, \rho)$ .
<b>elliptic_integral_RC</b>	Evaluates an elementary integral from which inverse circular functions, logarithms, and inverse hyperbolic functions can be computed.

FRESNEL INTEGRALS	
FUNCTION	DESCRIPTION
<a href="#">fresnel_integral_C</a>	Evaluates the cosine Fresnel integral.
<a href="#">fresnel_integral_S</a>	Evaluates the sine Fresnel integral.

AIRY FUNCTIONS	
FUNCTION	DESCRIPTION
<a href="#">airy_Ai</a>	Evaluates the Airy function.
<a href="#">airy_Bi</a>	Evaluates the Airy function of the second kind.
<a href="#">airy_Ai_derivative</a>	Evaluates the derivative of the Airy function.
<a href="#">airy_Bi_derivative</a>	Evaluates the derivative of the Airy function of the second kind.

KELVIN FUNCTIONS	
FUNCTION	DESCRIPTION
<a href="#">kelvin_ber0</a>	Evaluates the Kelvin function of the first kind, $ber$ , of order zero.
<a href="#">kelvin_bei0</a>	Evaluates the Kelvin function of the first kind, $bei$ , of order zero.
<a href="#">kelvin_ker0</a>	Evaluates the Kelvin function of the second kind, $ker$ , of order zero.
<a href="#">kelvin_kei0</a>	Evaluates the Kelvin function of the second kind, $kei$ , of order zero.
<a href="#">kelvin_ber0_derivative</a>	Evaluates the derivative of the Kelvin function of the first kind, $ber$ , of order zero.

KELVIN FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<b>kelvin_bei0_derivative</b>	Evaluates the derivative of the Kelvin function of the first kind, $bei$ , of order zero.
<b>kelvin_ker0_derivative</b>	Evaluates the derivative of the Kelvin function of the second kind, $ker$ , of order zero.
<b>kelvin_kei0_derivative</b>	Evaluates the derivative of the Kelvin function of the second kind, $kei$ , of order zero.

STATISTICAL FUNCTIONS	
FUNCTION	DESCRIPTION
<b>normal_cdf</b>	Evaluates the standard normal (Gaussian) distribution function.
<b>normal_inverse_cdf</b>	Evaluates the inverse of the standard normal (Gaussian) distribution function.
<b>chi_squared_cdf</b>	Evaluates the chi-squared distribution function.
<b>chi_squared_inverse_cdf</b>	Evaluates the inverse of the chi-squared distribution function.
<b>F_cdf</b>	Evaluates the $F$ distribution function.
<b>F_inverse_cdf</b>	Evaluates the inverse of the $F$ distribution function.
<b>t_cdf</b>	Evaluates the Student's $t$ distribution function.
<b>t_inverse_cdf</b>	Evaluates the inverse of the Student's $t$ distribution function.
<b>gamma_cdf</b>	Evaluates the gamma distribution function.
<b>binomial_cdf</b>	Evaluates the binomial distribution function.

STATISTICAL FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<b>hypergeometric_cdf</b>	Evaluates the hypergeometric distribution function.
<b>poisson_cdf</b>	Evaluates the Poisson distribution function.
<b>beta_cdf</b>	Evaluates the beta distribution function.
<b>beta_inverse_cdf</b>	Evaluates the inverse of the beta distribution function.
<b>bivariate_normal_cdf</b>	Evaluates the bivariate normal distribution function.

FINANCIAL FUNCTIONS	
FUNCTION	DESCRIPTION
<b>cumulative_interest</b>	Evaluates the cumulative interest paid between two periods.
<b>cumulative_principal</b>	Evaluates the cumulative principal paid between two periods.
<b>depreciation_db</b>	Evaluates the depreciation of an asset using the fixed-declining balance method.
<b>depreciation_ddb</b>	Evaluates the depreciation of an asset using the double-declining balance method.
<b>depreciation_sln</b>	Evaluates the depreciation of an asset using the straight-line method.
<b>depreciation_syd</b>	Evaluates the depreciation of an asset using the sum-of-years digits method.
<b>depreciation_vdb</b>	Evaluates the depreciation of an asset for any given period using the variable-declining balance method.
<b>dollar_decimal</b>	Converts a fractional price to a decimal price.

FINANCIAL FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<b>dollar_fraction</b>	Converts a decimal price to a fractional price.
<b>effective_rate</b>	Evaluates the effective annual interest rate.
<b>future_value</b>	Evaluates an investment's future value.
<b>future_value_schedule</b>	Evaluates the future value of an initial principal taking into consideration a schedule of compound interest rates.
<b>interest_payment</b>	Evaluates the interest payment for an investment for a given period.
<b>interest_rate_annuity</b>	Evaluates an annuity's interest rate per period.
<b>internal_rate_of_return</b>	Evaluates the internal rate of return for a schedule of cash flows.
<b>internal_rate_schedule</b>	Evaluates the internal rate of return for a schedule of cash flows. It is not necessary that the cash flows be periodic.
<b>modified_internal_rate</b>	Evaluates the modified internal rate of return for a schedule of periodic cash flows.
<b>net_present_value</b>	Evaluates an investment's net present value. The calculation is based on a schedule of periodic cash flows and a discount rate.
<b>nominal_rate</b>	Evaluates the nominal annual interest rate.
<b>number_of_periods</b>	Evaluates the number of periods for an investment for which periodic and constant payments are made and the interest rate is constant.
<b>payment</b>	Evaluates the periodic payment for an investment.
<b>present_value</b>	Evaluates the net present value of a stream of equal periodic cash flows, which are subject to a given discount rate.



FINANCIAL FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<b>present_value_schedule</b>	Evaluates the present value for a schedule of cash flows. It is not necessary that the cash flows be periodic.
<b>principal_payment</b>	Evaluates the payment on the principal for a specified period.

BOND FUNCTIONS	
FUNCTION	DESCRIPTION
<b>accr_interest_maturity</b>	Evaluates the interest that has accrued on a security, which pays interest at maturity.
<b>accr_interest_periodic</b>	Evaluates the interest that has accrued on a security, which pays interest periodically.
<b>bond_equivalent_yield</b>	Evaluates a Treasury bill's bond-equivalent yield.
<b>convexity</b>	Evaluates the convexity for a security.
<b>coupon_days</b>	Evaluates the number of days in the coupon period containing the settlement date.
<b>coupon_number</b>	Evaluates the number of coupons payable between the settlement date and the maturity date.
<b>days_before_settlement</b>	Evaluates the number of days starting with the beginning of the coupon period and ending with the settlement date.
<b>days_to_next_coupon</b>	Evaluates the number of days starting with the settlement date and ending with the next coupon date.
<b>depreciation_amordegrc</b>	Evaluates the depreciation for each accounting period.
<b>depreciation_amorlinc</b>	Evaluates the depreciation for each accounting period. This function is similar to <code>depreciation_amordegrc</code> , except that <code>depreciation_amordegrc</code> has a depreciation coefficient that is applied during the evaluation that is based on the asset life.

BOND FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<b>discount_price</b>	Evaluates a discounted security's price per \$100 face value.
<b>discount_rate</b>	Evaluates a security's discount rate.
<b>discount_yield</b>	Evaluates a discounted security's annual yield.
<b>duration</b>	Evaluates a security's annual duration where the security has periodic interest payments.
<b>interest_rate_security</b>	Evaluates a fully invested security's interest rate.
<b>modified_duration</b>	Evaluates a security's modified Macauley duration assuming a par value of \$100.
<b>next_coupon_date</b>	Evaluates the first coupon date that follows the settlement date.
<b>previous_coupon_date</b>	Evaluates the coupon date that immediately precedes the settlement date.
<b>price</b>	Evaluates a security's price per \$100 face value. The security pays periodic interest.
<b>price_maturity</b>	Evaluates a security's price per \$100 face value. The security pays interest at maturity.
<b>received_maturity</b>	Evaluates the amount one receives when a fully invested security reaches the maturity date.
<b>treasury_bill_price</b>	Evaluates a Treasury bill's price per \$100 face value.
<b>treasury_bill_yield</b>	Evaluates a Treasury bill's yield.
<b>year_fraction</b>	Evaluates the fraction of a year represented by the number of whole days between two dates.

BOND FUNCTIONS (Continued)	
FUNCTION	DESCRIPTION
<code>yield_maturity</code>	Evaluates a security's annual yield. The security pays interest at maturity.
<code>yield_periodic</code>	Evaluates a security's yield. The security pays periodic interest.

## CHAPTER 10: STATISTICS AND RANDOM NUMBER GENERATION

STATISTICS	
FUNCTION	DESCRIPTION
<a href="#">simple_statistics</a>	Computes basic univariate statistics.
<a href="#">table_oneway</a>	Tallies observations into a one-way frequency table.
<a href="#">chi_squared_test</a>	Performs a chi-squared goodness-of-fit test.
<a href="#">covariances</a>	Computes the sample variance-covariance or correlation matrix.
<a href="#">regression</a>	Fits a multiple linear regression model using least squares.
<a href="#">poly_regression</a>	Performs a polynomial least-squares regression.
<a href="#">ranks</a>	Computes the ranks, normal scores, or exponential scores for a vector of observations.

RANDOM NUMBERS	
FUNCTION	DESCRIPTION
<a href="#">random_seed_get</a>	Retrieves the current value of the seed used in the IMSL random number generators.
<a href="#">random_seed_set</a>	Initializes a random seed for use in the IMSL random number generators.

RANDOM NUMBERS (CONTINUED)	
FUNCTION	DESCRIPTION
<b>random_option</b>	Selects the uniform (0,1) multiplicative congruential pseudorandom number generator.
<b>random_uniform</b>	Generates pseudorandom numbers from a uniform (0,1) distribution.
<b>random_normal</b>	Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.
<b>random_poisson</b>	Generates pseudorandom numbers from a Poisson distribution.
<b>random_gamma</b>	Generates pseudorandom numbers from a standard gamma distribution.
<b>random_beta</b>	Generates pseudorandom numbers from a beta distribution.
<b>random_exponential</b>	Generates pseudorandom numbers from a standard exponential distribution.
<b>faure_next_point</b>	Computes a shuffled Faure sequence.

---

## CHAPTER 11: PRINTING FUNCTIONS

PRINT	
FUNCTION	DESCRIPTION
<code>write_matrix</code>	Prints a rectangular matrix (or vector) stored in contiguous memory locations.
<code>page</code>	Sets or retrieves the page width or length.
<code>write_options</code>	Sets or retrieves an option for printing a matrix.

## CHAPTER 12: UTILITIES

### SET OUTPUT FILES

FUNCTION	DESCRIPTION
<b>output_file</b>	Sets the output file or the error message output file.
<b>version</b>	Returns integer information describing the version of the library, license number, operating system, and compiler.

### TIME AND DATE

FUNCTION	DESCRIPTION
<b>ctime</b>	Returns the number of CPU seconds used.
<b>date_to_days</b>	Computes the number of days from January 1, 1900, to the given date.
<b>days_to_date</b>	Gives the date corresponding to the number of days since January 1, 1900.

### ERROR HANDLING

FUNCTION	DESCRIPTION
<b>error_options</b>	Sets various error handling options.
<b>error_type</b>	Gets the type corresponding to the error message from the last function called.
<b>error_message</b>	Gets the text of the error message from the last function called.

**ERROR HANDLING (CONTINUED)**

FUNCTION	DESCRIPTION
<b>error_code</b>	Gets the code corresponding to the error message from the last function called.
<b>initialize_error_handler</b>	Initializes the IMSL C Numerical Library error handling system.
<b>set_user_fcn_return_flag</b>	Indicates a condition has occurred in a user-supplied function necessitating a return to the calling C Numerical Library function.

**C RUNTIME LIBRARY**

FUNCTION	DESCRIPTION
<b>free</b>	Frees memory returned from an IMSL C Math Library function.
<b>fopen</b>	Opens a file using the C runtime library used by the IMSL C Math Library.
<b>fclose</b>	Closes a file opened by <code>ims1_fopen</code> .

**OPEN MP**

FUNCTION	DESCRIPTION
<b>omp_options</b>	Sets various OpenMP options.

**CONSTANTS**

FUNCTION	DESCRIPTION
<b>constant</b>	Returns the value of various mathematical and physical constants.



CONSTANTS (CONTINUED)	
FUNCTION	DESCRIPTION
<b>machine (float)</b>	Returns information describing the computer's floating-point arithmetic.
<b>machine (integer)</b>	Returns integer information describing the computer's arithmetic.
<b>sort</b>	Sorts a vector by algebraic value. Optionally, a vector can be sorted by absolute value, and a sort permutation can be returned.
<b>sort_integer</b>	Sorts an integer vector by algebraic value. Optionally, a vector can be sorted by absolute value, and a sort permutation can be returned.

COMPUTING VECTOR NORMS	
FUNCTION	DESCRIPTION
<b>vector_norm</b>	Computes various norms of a vector or the difference of two vectors.
<b>vector_norm (complex)</b>	Computes various norms of a vector or the difference of two vectors

LINEAR ALGEBRA SUPPORT	
FUNCTION	DESCRIPTION
<b>mat_mul_rect</b>	Computes the transpose of a matrix, a matrix-vector product, a matrix-matrix product, the bilinear form, or any triple product.
<b>mat_mul_rect (complex)</b>	Computes the transpose of a matrix, the conjugate-transpose of a matrix, a matrix-vector product, a matrix-matrix product, the bilinear form, or any triple product.
<b>mat_mul_rect_band</b>	Computes the transpose of a matrix, a matrix-vector product, or a matrix-matrix product, all matrices stored in band form.
<b>mat_mul_rect_band (complex)</b>	Computes the transpose of a matrix, a matrix-vector product, or a matrix-matrix product, all matrices of complex type and stored in band form.

LINEAR ALGEBRA SUPPORT (CONTINUED)	
FUNCTION	DESCRIPTION
<b>mat_mul_rect_coordinate</b>	Computes the transpose of a matrix, a matrix-vector product, or a matrix-matrix product, all matrices stored in sparse coordinate form.
<b>mat_mul_rect_coordinate (complex)</b>	Computes the transpose of a matrix, a matrix-vector product or a matrix-matrix product, all matrices stored in sparse coordinate form.
<b>mat_add_band</b>	Adds two band matrices, both in band storage mode, $C \leftarrow \alpha A + \beta B$ .
<b>mat_add_band (complex)</b>	Adds two band complex matrices, both in band storage mode, $C \leftarrow \alpha A + \beta B$ .
<b>mat_add_coordinate</b>	Performs element-wise addition of two real matrices stored in coordinate format, $C \leftarrow \alpha A + \beta B$ .
<b>mat_add_coordinate (complex)</b>	Performs element-wise addition on two complex matrices stored in coordinate format, $C \leftarrow \alpha A + \beta B$ .
<b>matrix_norm</b>	Computes various norms of a rectangular matrix.
<b>matrix_norm_band</b>	Computes various norms of a matrix stored in band storage mode.
<b>matrix_norm_coordinate</b>	Computes various norms of a matrix stored in coordinate format.
<b>generate_test_band</b>	Generates test matrices of class $E(n, c)$ . Returns in band or band symmetric format.
<b>generate_test_band (complex)</b>	Generates test matrices of class $E_c(n, c)$ . Returns in band or band symmetric format
<b>generate_test_coordinate</b>	Generates test matrices of class $D(n, c)$ and $E(n, c)$ . Returns in either coordinate format.
<b>generate_test_coordinate (complex)</b>	Generates test matrices of class $D(n, c)$ and $E(n, c)$ . Returns in either coordinate or band storage format, where possible.

GPU SUPPORT	
FUNCTION	DESCRIPTION
<code>cuda_get</code>	Gets the threshold used by the specified function to determine if the <i>NVIDIA® CUDA™ Toolkit</i> algorithm will be used.
<code>cuda_set</code>	Sets the threshold used by the specified function to determine if the <i>NVIDIA® CUDA™ Toolkit</i> algorithm will be used.
<code>cuda_free</code>	Releases NVIDIA memory allocated by the IMSL C Numerical Library.

NUMERIC UTILITIES	
FUNCTION	DESCRIPTION
<code>c_neg</code>	Changes the sign of a complex number.
<code>c_add</code>	Adds two complex numbers.
<code>c_sub</code>	Subtracts a complex number from a complex number.
<code>c_mul</code>	Multiplies two complex numbers.
<code>c_div</code>	Divides a complex number by a complex number.
<code>c_eq</code>	Tests for equality of two complex numbers.
<code>cz_convert</code>	Truncates a double precision complex number to a single precision complex number.
<code>zc_convert</code>	Increases precision of a single precision complex number to a double precision complex number.

## NUMERIC UTILITIES (CONTINUED)

FUNCTION	DESCRIPTION
<b>cf_convert</b>	Makes a complex number from an ordered pair.
<b>c_conjg</b>	Conjugates a complex number.
<b>c_abs</b>	Computes a magnitude of a complex number.
<b>c_arg</b>	Computes an angle of a complex number.
<b>c_sqrt</b>	Computes a square root of a complex number.
<b>c_cos</b>	Computes a trigonometric cosine of a complex number.
<b>c_sin</b>	Computes a trigonometric sine of a complex number.
<b>c_exp</b>	Computes an exponential function of a complex number.
<b>c_log</b>	Computes a natural logarithm of a complex number.
<b>cf_power</b>	Computes a complex number raised to a real power.
<b>cc_power</b>	Computes a complex number raised to a complex power.
<b>fi_power</b>	Computes a real number raised to an integral power.
<b>ii_power</b>	Computes an integer raised to an integral power.

# IMSL C STAT LIBRARY

## CHAPTER 1: BASIC STATISTICS

### SIMPLE SUMMARY STATISTICS

FUNCTION	DESCRIPTION
<code>simple_statistics</code>	Computes basic univariate statistics.
<code>empirical_quantiles</code>	Computes empirical quantiles.
<code>normal_one_sample</code>	Computes statistics for mean and variance inferences using a sample from a normal population.
<code>normal_two_sample</code>	Computes statistics for mean and variance inferences using samples from two normal populations.

### TABULATE, SORT, AND RANK

FUNCTION	DESCRIPTION
<code>table_oneway</code>	Tallies observations into a one-way frequency table.
<code>table_twoway</code>	Tallies observations into a two-way frequency table.
<code>sort_data</code>	Sorts observations by specified keys, with option to tally cases into a multi-way frequency table.
<code>ranks</code>	Computes the ranks, normal scores, or exponential scores for a vector of observations.

## CHAPTER 2: REGRESSION

### MULTIVARIATE LINEAR REGRESSION—MODEL FITTING

FUNCTION	DESCRIPTION
<code>regressors_for_glm</code>	Generates regressors for a general linear model.
<code>regression</code>	Fits a multiple linear regression model using least squares.

### MULTIVARIATE LINEAR REGRESSION — STATISTICAL INFERENCE AND DIAGNOSTICS

FUNCTION	DESCRIPTION
<code>regression_summary</code>	Produces summary statistics for a regression model given the information from the fit.
<code>regression_prediction</code>	Computes predicted values, confidence intervals, and diagnostics after fitting a regression model.
<code>hypothesis_partial</code>	Constructs a completely testable hypothesis.
<code>hypothesis_scph</code>	Sums of cross products for a multivariate hypothesis.
<code>hypothesis_test</code>	Tests for the multivariate linear hypothesis.

VARIABLE SELECTION	
FUNCTION	DESCRIPTION
<a href="#">regression_selection</a>	Selects the best multiple linear regression models.
<a href="#">regression_stepwise</a>	Builds multiple linear regression models using forward selection, backward selection or stepwise selection.

POLYNOMIAL AND NONLINEAR REGRESSION	
FUNCTION	DESCRIPTION
<a href="#">poly_regression</a>	Performs a polynomial least-squares regression.
<a href="#">poly_prediction</a>	Computes predicted values, confidence intervals, and diagnostics after fitting a polynomial regression model.
<a href="#">nonlinear_regression</a>	Fits a nonlinear regression model.
<a href="#">nonlinear_optimization</a>	Fits a nonlinear regression model using Powell's algorithm.

ALTERNATIVES TO LEAST SQUARES	
FUNCTION	DESCRIPTION
<a href="#">Lnorm_regression</a>	Fits a multiple linear regression model using $L_p$ criteria other than least squares.
<a href="#">pls_regression</a>	Performs partial least squares (PLS) regression for one or more response variables and one or more predictor variables.

---

## CHAPTER 3: CORRELATION AND COVARIANCE

### VARIANCES, COVARIANCES, AND CORRELATIONS

FUNCTION	DESCRIPTION
<b>covariances</b>	Computes the variance-covariance or correlation matrix.
<b>partial_covariances</b>	Computes a pooled variance-covariance matrix from the observations.
<b>pooled_covariances</b>	Computes partial correlations or covariances from the covariance or correlation matrix.
<b>robust_covariances</b>	Computes a robust estimate of a covariance matrix and mean vector.



## CHAPTER 4: ANALYSIS OF VARIANCE AND DESIGNED EXPERIMENTS

### GENERAL ANALYSIS

FUNCTION	DESCRIPTION
<a href="#">anova_oneway</a>	Analyzes a one-way classification model.
<a href="#">ancovar</a>	Analyzes a one-way classification model with covariates.
<a href="#">anova_factorial</a>	Analyzes a balanced factorial design with fixed effects.
<a href="#">anova_nested</a>	Analyzes a completely nested random effects model with possibly unequal numbers in the subgroups.
<a href="#">anova_balanced</a>	Analyzes a balanced complete experimental design for a fixed, random, or mixed model.

### DESIGNED EXPERIMENTS

FUNCTION	DESCRIPTION
<a href="#">crd_factorial</a>	Analyzes data from balanced and unbalanced completely randomized experiments.
<a href="#">rcbd_factorial</a>	Analyzes data from balanced and unbalanced randomized complete-block experiments.
<a href="#">latin_square</a>	Analyzes data from latin-square experiments.
<a href="#">lattice</a>	Analyzes balanced and partially-balanced lattice experiments.

DESIGNED EXPERIMENTS (CONTINUED)	
FUNCTION	DESCRIPTION
<b>split_plot</b>	Analyzes a wide variety of split-plot experiments with fixed, mixed or random factors.
<b>split_split_plot</b>	Analyzes data from split-split-plot experiments.
<b>strip_plot</b>	Analyzes data from strip-plot experiments.
<b>strip_split_plot</b>	Analyzes data from strip-split-plot experiments.

UTILITIES	
FUNCTION	DESCRIPTION
<b>homogeneity</b>	Conducts Bartlett's and Levene's tests of the homogeneity of variance assumption in analysis of variance.
<b>multiple_comparisons</b>	Compares differences among averages using the SNK, LSD, Tukey's, Duncan's and Bonferroni's multiple comparisons tests.
<b>false_discovery_rates</b>	Calculates the False Discovery Rate (FDR) $q$ -values corresponding to a set of $p$ -values resulting from multiple simultaneous hypothesis tests.
<b>yates</b>	Estimates missing observations in designed experiments using Yate's method.

## CHAPTER 5: CATEGORICAL AND DISCRETE DATA ANALYSIS

### STATISTICS IN THE TWO-WAY CONTINGENCY TABLE

FUNCTION	DESCRIPTION
<a href="#">contingency_table</a>	Performs a chi-squared analysis of a two-way contingency table.
<a href="#">exact_enumeration</a>	Computes exact probabilities in a two-way contingency table, using the total enumeration method.
<a href="#">exact_network</a>	Computes exact probabilities in a two-way contingency table using the network algorithm.

### CATEGORICAL MODELS

FUNCTION	DESCRIPTION
<a href="#">categorical_glm</a>	Analyzes categorical data using logistic, Probit, Poisson, and other generalized linear models.
<a href="#">logistic_regression</a>	Fits a binomial or multinomial logistic regression model using iteratively reweighted least squares.
<a href="#">logistic_reg_predict</a>	Predicts a binomial or multinomial outcome given an estimated model and new values of the independent variables.

## CHAPTER 6: NONPARAMETRIC STATISTICS

### ONE SAMPLE TESTS-NONPARAMETRIC STATISTICS

FUNCTION	DESCRIPTION
<b>sign_test</b>	Performs a sign test.
<b>wilcoxon_sign_rank</b>	Performs a Wilcoxon signed rank test.
<b>noether_cyclical_trend</b>	Performs the Noether's test for cyclical trend.
<b>cox_stuart_trends_test</b>	Performs the Cox and Stuart sign test for trends in location and dispersion.
<b>tie_statistics</b>	Computes tie statistics for a sample of observations.

### TWO OR MORE SAMPLES

FUNCTION	DESCRIPTION
<b>wilcoxon_rank_sum</b>	Performs a Wilcoxon rank sign test.
<b>kruskal_wallis_test</b>	Performs a Kruskal-Wallis test for identical population medians.
<b>friedmans_test</b>	Performs Friedman's test for a randomized complete block design.
<b>cochran_q_test</b>	Performs Cochran's Q test for related observations.
<b>k_trends_test</b>	Performs $k$ -sample trends test against ordered alternatives.

## CHAPTER 7: TESTS OF GOODNESS OF FIT

### GENERAL GOODNESS-OF-FIT TESTS FOR A SPECIFIED DISTRIBUTION

FUNCTION	DESCRIPTION
<a href="#">chi_squared_test</a>	Performs a chi-squared goodness-of-fit test.
<a href="#">shapiro_wilk_normality_test</a>	Performs the Shapiro-Wilk test for normality.
<a href="#">lilliefors_normality_test</a>	Performs a Lilliefors test for normality.
<a href="#">chi_squared_normality_test</a>	Performs a chi-squared test for normality.
<a href="#">kolmogorov_one</a>	Performs a Kolmogorov-Smirnov's one-sample test for continuous distributions.
<a href="#">kolmogorov_two</a>	Performs a Kolmogorov-Smirnov's two-sample test.
<a href="#">multivar_normality_test</a>	Computes Mardia's multivariate measures of skewness and kurtosis and tests for multivariate normality.
<a href="#">ad_normality_test</a>	Performs an Anderson-Darling test for normality.
<a href="#">cvm_normality_test</a>	Performs a Cramer-von Mises test for normality.

### TESTS FOR RANDOMNESS

FUNCTION	DESCRIPTION
<a href="#">randomness_test</a>	Performs a test for randomness.

## CHAPTER 8: TIME SERIES ANALYSIS AND FORECASTING

ARIMA MODELS	
FUNCTION	DESCRIPTION
<b>arma</b>	Computes least-square estimates of parameters for an ARMA model.
<b>max_arma</b>	Exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive, moving average) time series model.
<b>arma_forecast</b>	Computes forecasts and their associated probability limits for an ARMA model.
<b>arima</b>	Fits a univariate seasonal or non-seasonal ARIMA time series model.
<b>regression_arima</b>	Fits a univariate ARIMA ( $p, d, q$ ) time series model with the inclusion of one or more regression variables.

AUTOMATIC ARIMA SELECTION AND FITTING UTILITIES	
FUNCTION	DESCRIPTION
<b>auto_uni_ar</b>	Automatic selection and fitting of a univariate autoregressive time series model. The lag for the model is automatically selected using Akaike's information criterion (AIC).
<b>seasonal_fit</b>	Estimates the optimum seasonality parameters for a time series using an autoregressive model, $AR(p)$ , to represent the time series.
<b>ts_outlier_identification</b>	Detects and determines outliers and simultaneously estimates the model parameters in a time series whose underlying outlier-free series follows a general seasonal or nonseasonal ARMA model.
<b>ts_outlier_forecast</b>	Computes forecasts, their associated probability limits and weights for an outlier contaminated time series whose underlying outlier free series follows a general seasonal or nonseasonal ARMA model.

**AUTOMATIC ARIMA SELECTION AND FITTING UTILITIES (CONTINUED)**

FUNCTION	DESCRIPTION
<b>auto_arima</b>	Automatically identifies time series outliers, determines parameters of a multiplicative seasonal ARIMA $(p,0,q) \times (0,d,0)_S$ model and produces forecasts that incorporate the effects of outliers whose effects persist beyond the end of the series.
<b>auto_parm</b>	Estimates structural breaks in non-stationary univariate time series.

**BAYESIAN TIME SERIES ESTIMATION**

FUNCTION	DESCRIPTION
<b>bayesian_seasonal_adj</b>	Decomposes a time series into trend, seasonal, and an error component.

**MODEL CONSTRUCTION AND EVALUATION UTILITIES**

FUNCTION	DESCRIPTION
<b>box_cox_transform</b>	Performs a Box-Cox transformation.
<b>difference</b>	Differences a seasonal or nonseasonal time series.
<b>autocorrelation</b>	Computes the sample autocorrelation function of a stationary time series.
<b>crosscorrelation</b>	Computes the sample cross-correlation function of two stationary time series.
<b>multi_crosscorrelation</b>	Computes the multichannel cross-correlation function of two mutually stationary multichannel time series.
<b>partial_autocorrelation</b>	Computes the sample partial autocorrelation function of a stationary time series

### MODEL CONSTRUCTION AND EVALUATION UTILITIES (CONTINUED)

FUNCTION	DESCRIPTION
<b>lack_of_fit</b>	Performs lack-of-fit test for an univariate time series or transfer function given the appropriate correlation function.
<b>estimate_missing</b>	Estimates missing values in a time series.

### EXPONENTIAL SMOOTHING METHODS

FUNCTION	DESCRIPTION
<b>hw_time_series</b>	Calculates parameters and forecasts using the Holt-Winters Multiplicative or Additive forecasting method for seasonal data.

### GARCH MODELING

FUNCTION	DESCRIPTION
<b>garch</b>	Computes estimates of the parameters of a GARCH( $p,q$ ) model.

### STATE-SPACE MODELS

FUNCTION	DESCRIPTION
<b>kalman</b>	Performs Kalman filtering and evaluates the likelihood function for the state-space model.

### AUTO-REGRESSION AND ERROR CORRECTION

FUNCTION	DESCRIPTION
<b>vector_autoregression</b>	Estimates a vector auto-regressive time series model with optional moving average components.



## CHAPTER 9: MULTIVARIATE ANALYSIS

### HIERARCHICAL CLUSTER ANALYSIS

FUNCTION	DESCRIPTION
<b>dissimilarities</b>	Computes a matrix of dissimilarities (or similarities) between the columns (or rows) of a matrix.
<b>cluster_hierarchical</b>	Performs a hierarchical cluster analysis given a distance matrix.
<b>cluster_number</b>	Computes cluster membership for a hierarchical cluster tree.

### K-MEANS CLUSTER ANALYSIS

FUNCTION	DESCRIPTION
<b>cluster_k_means</b>	Performs a <i>K</i> -means (centroid) cluster analysis.

### PRINCIPAL COMPONENT ANALYSIS

FUNCTION	DESCRIPTION
<b>principal_components</b>	Computes principal components.

### FACTOR ANALYSIS

FUNCTION	DESCRIPTION
<b>factor_analysis</b>	Extracts initial factor-loading estimates in factor analysis with rotation options.

FACTOR ANALYSIS (CONTINUED)

FUNCTION	DESCRIPTION
<a href="#">discriminant_analysis</a>	Performs discriminant function analysis.

## CHAPTER 10: SURVIVAL AND RELIABILITY ANALYSIS

### SURVIVAL ANALYSIS

FUNCTION	DESCRIPTION
<code>kaplan_meier_estimates</code>	Computes Kaplan-Meier estimates of survival probabilities in stratified samples.
<code>prop_hazards_gen_lin</code>	Analyzes survival and reliability data using Cox's proportional hazards model.
<code>survival_glm</code>	Analyzes censored survival data using a generalized linear model.
<code>survival_estimates</code>	Estimates survival probabilities and hazard rates for the various parametric models.

### RELIABILITY ANALYSIS

FUNCTION	DESCRIPTION
<code>nonparam_hazard_rate</code>	Estimates a reliability hazard function using a nonparametric approach.

### ACTUARIAL TABLES

FUNCTION	DESCRIPTION
<code>life_tables</code>	Produces population and cohort life tables.

## CHAPTER 11: PROBABILITY DISTRIBUTION FUNCTIONS AND INVERSES

### DISCRETE RANDOM VARIABLES

FUNCTION	DESCRIPTION
<b>binomial_cdf</b>	Evaluates the binomial distribution function.
<b>binomial_pdf</b>	Evaluates the binomial probability function.
<b>geometric_cdf</b>	Evaluates the discrete geometric cumulative distribution function.
<b>geometric_inverse_cdf</b>	Evaluates the inverse of the discrete geometric cumulative distribution function.
<b>geometric_pdf</b>	Evaluates the discrete geometric probability density function.
<b>hypergeometric_cdf</b>	Evaluates the hypergeometric distribution function.
<b>hypergeometric_pdf</b>	Evaluates the hypergeometric probability function.
<b>poisson_cdf</b>	Evaluates the Poisson distribution function.
<b>poisson_pdf</b>	Evaluates the Poisson probability function.
<b>discrete_uniform_cdf</b>	Evaluates the discrete uniform cumulative distribution function.
<b>discrete_uniform_inverse_cdf</b>	Evaluates the inverse of the discrete uniform cumulative distribution function.
<b>discrete_uniform_pdf</b>	Evaluates the discrete uniform probability density function .

## CONTINUOUS RANDOM VARIABLES

FUNCTION	DESCRIPTION
<b>beta_cdf</b>	Evaluates the beta probability distribution function.
<b>beta_inverse_cdf</b>	Evaluates the inverse of the beta distribution function.
<b>non_central_beta_cdf</b>	Evaluates the noncentral beta cumulative distribution function.
<b>non_central_beta_inverse_cdf</b>	Evaluates the inverse of the noncentral beta cumulative distribution function.
<b>non_central_beta_pdf</b>	Evaluates the noncentral beta probability density function.
<b>bivariate_normal_cdf</b>	Evaluates the bivariate normal distribution function.
<b>chi_squared_cdf</b>	Evaluates the chi-squared distribution function.
<b>chi_squared_inverse_cdf</b>	Evaluates the inverse of the chi-squared distribution function.
<b>complementary_chi_squared_cdf</b>	Evaluates the complement of the chi-squared distribution.
<b>complementary_F_cdf</b>	Evaluates the complement of the $F$ distribution function.
<b>complementary_t_cdf</b>	Evaluates the complement of the Student's $t$ distribution function.
<b>exponential_cdf</b>	Evaluates the exponential cumulative distribution function.
<b>exponential_inverse_cdf</b>	Evaluates the inverse of the exponential cumulative distribution function.

CONTINUOUS RANDOM VARIABLES (CONTINUED)	
FUNCTION	DESCRIPTION
<b>exponential_pdf</b>	Evaluates the exponential probability density function.
<b>F_cdf</b>	Evaluates the $F$ distribution function.
<b>F_inverse_cdf</b>	Evaluates the inverse of the $F$ distribution function.
<b>gamma_cdf</b>	Evaluates the gamma distribution function.
<b>gamma_inverse_cdf</b>	Evaluates the inverse of the gamma distribution function.
<b>lognormal_cdf</b>	Evaluates the lognormal cumulative distribution function.
<b>lognormal_inverse_cdf</b>	Evaluates the inverse of the lognormal cumulative distribution function.
<b>lognormal_pdf</b>	Evaluates the lognormal probability density function.
<b>multivariate_normal_cdf</b>	Evaluates the cumulative distribution function for the multivariate normal distribution.
<b>non_central_chi_sq</b>	Evaluates the noncentral chi-squared distribution function.
<b>non_central_chi_sq_inv</b>	Evaluates the inverse of the noncentral chi-squared function.
<b>non_central_chi_sq_pdf</b>	Evaluates the noncentral chi-squared probability density function.
<b>non_central_F_cdf</b>	Evaluates the noncentral $F$ cumulative distribution function.
<b>complementary_non_central_F_cdf</b>	Evaluates the complementary noncentral $F$ cumulative distribution function.

**CONTINUOUS RANDOM VARIABLES (CONTINUED)**

FUNCTION	DESCRIPTION
<a href="#">non_central_F_inverse_cdf</a>	Evaluates the inverse of the noncentral $F$ cumulative distribution function.
<a href="#">non_central_F_pdf</a>	Evaluates the noncentral $F$ probability density function.
<a href="#">non_central_t_cdf</a>	Evaluates the noncentral Student's $t$ distribution function.
<a href="#">non_central_t_inv_cdf</a>	Evaluates the inverse of the noncentral Student's $t$ distribution function.
<a href="#">non_central_t_pdf</a>	Evaluates the noncentral Student's $t$ probability density function.
<a href="#">pareto_cdf</a>	Evaluates the Pareto cumulative probability distribution function.
<a href="#">pareto_pdf</a>	Evaluates the Pareto probability density function.
<a href="#">normal_cdf</a>	Evaluates the standard normal (Gaussian) distribution function.
<a href="#">normal_inverse_cdf</a>	Evaluates the inverse of the standard normal (Gaussian) distribution function.
<a href="#">t_cdf</a>	Evaluates the Student's $t$ distribution function.
<a href="#">t_inverse_cdf</a>	Evaluates the inverse of the Student's $t$ distribution function.

**PARAMETER ESTIMATION**

FUNCTION	DESCRIPTION
<a href="#">max_likelihood_estimates</a>	Calculates maximum likelihood estimates (MLE) for the parameters of one of several univariate probability distributions.

## CHAPTER 12: RANDOM NUMBER GENERATION

### UNIVARIATE DISCRETE DISTRIBUTIONS

FUNCTION	DESCRIPTION
<b>random_binomial</b>	Generates pseudorandom binomial numbers from a binomial distribution.
<b>random_geometric</b>	Generates pseudorandom numbers from a geometric distribution.
<b>random_hypergeometric</b>	Generates pseudorandom numbers from a hypergeometric distribution.
<b>random_logarithmic</b>	Generates pseudorandom numbers from a logarithmic distribution.
<b>random_neg_binomial</b>	Generates pseudorandom numbers from a negative binomial distribution.
<b>random_poisson</b>	Generates pseudorandom numbers from a Poisson distribution.
<b>random_uniform_discrete</b>	Generates pseudorandom numbers from a discrete uniform distribution.
<b>random_general_discrete</b>	Generates pseudorandom numbers from a general discrete distribution using an alias method or, optionally, a table lookup method.
<b>discrete_table_setup</b>	Sets up a table to generate pseudorandom numbers from a general discrete distribution.

### UNIVARIATE CONTINUOUS DISTRIBUTIONS

FUNCTION	DESCRIPTION
<b>random_beta</b>	Generates pseudorandom numbers from a beta distribution.



## UNIVARIATE CONTINUOUS DISTRIBUTIONS (CONTINUED)

FUNCTION	DESCRIPTION
<b>random_cauchy</b>	Generates pseudorandom numbers from a Cauchy distribution.
<b>random_chi_squared</b>	Generates pseudorandom numbers from a chi-squared distribution.
<b>random_exponential</b>	Generates pseudorandom numbers from a standard exponential distribution.
<b>random_exponential_mix</b>	Generates pseudorandom mixed numbers from a standard exponential distribution.
<b>random_gamma</b>	Generates pseudorandom numbers from a standard gamma distribution.
<b>random_lognormal</b>	Generates pseudorandom numbers from a lognormal distribution.
<b>random_normal</b>	Generates pseudorandom numbers from a standard normal distribution.
<b>random_stable</b>	Generates pseudorandom numbers from a stable distribution.
<b>random_student_t</b>	Generates pseudorandom numbers from a Student's $t$ distribution.
<b>random_triangular</b>	Generates pseudorandom numbers from a triangular distribution.
<b>random_uniform</b>	Generates pseudorandom numbers from a uniform (0, 1) distribution.
<b>random_von_mises</b>	Generates pseudorandom numbers from a von Mises distribution.
<b>random_weibull</b>	Generates pseudorandom numbers from a Weibull distribution.
<b>random_general_continuous</b>	Generates pseudorandom numbers from a general continuous distribution.

**UNIVARIATE CONTINUOUS DISTRIBUTIONS (CONTINUED)**

FUNCTION	DESCRIPTION
<a href="#">continuous_table_setup</a>	Sets up a table to generate pseudorandom numbers from a general continuous distribution.

**MULTIVARIATE CONTINUOUS DISTRIBUTIONS**

FUNCTION	DESCRIPTION
<a href="#">random_normal_multivariate</a>	Generates pseudorandom numbers from a multivariate normal distribution.
<a href="#">random_orthogonal_matrix</a>	Generates a pseudorandom orthogonal matrix or a correlation matrix.
<a href="#">random_mvar_from_data</a>	Generates pseudorandom numbers from a multivariate distribution determined from a given sample.
<a href="#">random_multinomial</a>	Generates pseudorandom numbers from a multinomial distribution.
<a href="#">random_sphere</a>	Generates pseudorandom points on a unit circle or $K$ -dimensional sphere.
<a href="#">random_table_twoway</a>	Generates a pseudorandom two-way table.
<a href="#">random_mvar_gaussian_copula</a>	Given a Cholesky factorization of a correlation matrix, generates pseudorandom numbers from a Gaussian Copula distribution.
<a href="#">random_mvar_t_copula</a>	Given a Cholesky factorization of a correlation matrix, generates pseudorandom numbers from a Student's $t$ Copula distribution.
<a href="#">canonical_correlation</a>	Given an input array of deviate values, generates a canonical correlation array.

ORDER STATISTICS	
FUNCTION	DESCRIPTION
<a href="#">random_order_normal</a>	Generates pseudorandom order statistics from a standard normal distribution.
<a href="#">random_order_uniform</a>	Generates pseudorandom order statistics from a uniform (0, 1) distribution.

STOCHASTIC PROCESSES	
FUNCTION	DESCRIPTION
<a href="#">random_arma</a>	Generates a time series from a specific ARMA model.
<a href="#">random_npp</a>	Generates pseudorandom numbers from a nonhomogeneous Poisson process.

SAMPLES AND PERMUTATIONS	
FUNCTION	DESCRIPTION
<a href="#">random_permutation</a>	Generates a pseudorandom permutation.
<a href="#">random_sample_indices</a>	Generates a simple pseudorandom sample of indices.
<a href="#">random_sample</a>	Generates a simple pseudorandom sample from a finite population.

UTILITY FUNCTIONS	
FUNCTION	DESCRIPTION
<b>random_option</b>	Selects the uniform (0, 1) multiplicative congruential pseudorandom number generator.
<b>random_option_get</b>	Retrieves the uniform (0, 1) multiplicative congruential pseudorandom number generator.
<b>random_seed_get</b>	Retrieves the current value of the seed used in the IMSL random number generators.
<b>random_substream_seed_get</b>	Retrieves a seed for the congruential generators that do not do shuffling that will generate random numbers beginning 100,000 numbers farther along.
<b>random_seed_set</b>	Initializes a random seed for use in the IMSL random number generators.
<b>random_table_set</b>	Sets the current table used in the shuffled generator.
<b>random_table_get</b>	Retrieves the current table used in the shuffled generator.
<b>random_GFSR_table_set</b>	Sets the current table used in the GFSR generator.
<b>random_GFSR_table_get</b>	Retrieves the current table used in the GFSR generator.
<b>random_MT32_init</b>	Initializes the 32-bit Mersenne Twister generator using an array.
<b>random_MT32_table_get</b>	Retrieves the current table used in the 32-bit Mersenne Twister generator.
<b>random_MT32_table_set</b>	Sets the current table used in the 32-bit Mersenne Twister generator.
<b>random_MT64_init</b>	Initializes the 64-bit Mersenne Twister generator using an array.

## UTILITY FUNCTIONS (CONTINUED)

FUNCTION	DESCRIPTION
<code>random_MT64_table_get</code>	Retrieves the current table used in the 64-bit Mersenne Twister generator.
<code>random_MT64_table_set</code>	Sets the current table used in the 64-bit Mersenne Twister generator.

## LOW-DISCREPANCY SEQUENCE

FUNCTION	DESCRIPTION
<code>faure_next_point</code>	Computes a shuffled Faure sequence.

## CHAPTER 13: DATA MINING

### APRIORI – MARKET BASKET ANALYSIS

FUNCTION	DESCRIPTION
<b>apriori</b>	Computes the frequent itemsets in a transaction set.
<b>aggr_apriori</b>	Computes the frequent itemsets in a transaction set using aggregation.
<b>write_apriori_itemsets</b>	Prints frequent itemsets.
<b>write_association_rules</b>	Prints association rules.
<b>free_apriori_itemsets</b>	Frees the memory allocated within a frequent itemsets structure.
<b>free_association_rules</b>	Frees the memory allocated within an association rules structure.

### DECISION TREE

FUNCTION	DESCRIPTION
<b>decision_tree</b>	Generates a decision tree or a random forest of decision trees for a single response variable and two or more predictor variables
<b>decision_tree_predict</b>	Computes predicted values using a decision tree.
<b>decision_tree_print</b>	Prints a decision tree.
<b>decision_tree_free</b>	Frees the memory associated with a decision tree.

DECISION TREE	
FUNCTION	DESCRIPTION
<b>gradient_boosting</b>	Performs stochastic gradient boosting for decision trees.

GENETIC ALGORITHM DATA STRUCTURES	
FUNCTION	DESCRIPTION
<b>ga_chromosome</b>	Creates an <i>Imsls_f_chromosome</i> data structure containing unencoded and encoded phenotype information.
<b>ga_copy_chromosome</b>	Copies the contents of one chromosome into another chromosome.
<b>ga_clone_chromosome</b>	Clones an existing chromosome.
<b>ga_individual</b>	Creates an <i>Imsls_f_individual</i> data structure from user supplied phenotypes.
<b>ga_copy_individual</b>	Copies the contents of one individual into another individual.
<b>ga_clone_individual</b>	Clones an existing individual.
<b>ga_mutate</b>	Performs the mutation operation on an individual's chromosome.
<b>ga_decode</b>	Decodes an individual's chromosome into its binary, nominal, integer and real phenotypes.
<b>ga_encode</b>	Encodes an individual's binary, nominal, integer and real phenotypes into its chromosome.
<b>ga_free_individual</b>	Frees memory allocated to an existing individual.
<b>ga_population</b>	Creates an <i>Imsls_f_population</i> data structure from user supplied individuals.

**GENETIC ALGORITHM DATA STRUCTURES (CONTINUED)**

FUNCTION	DESCRIPTION
<b>ga_random_population</b>	Creates an <i>Imsls_f_population</i> data structure from randomly generated individuals.
<b>ga_copy_population</b>	Copies the contents of one population into another population.
<b>ga_clone_population</b>	Clones an existing population.
<b>ga_grow_population</b>	Adds individuals to an existing population.
<b>ga_merge_population</b>	Creates a new population by merging two populations with identical chromosome structures.
<b>ga_free_population</b>	Frees memory allocated to an existing population.
<b>genetic_algorithm</b>	Optimizes a user defined fitness function using a tailored genetic algorithm.

**NAIVE BAYES**

FUNCTION	DESCRIPTION
<b>naive_bayes_trainer</b>	Trains a Naïve Bayes classifier.
<b>naive_bayes_classification</b>	Classifies unknown patterns using a previously trained Naïve Bayes classifier.
<b>nb_classifier_free</b>	Frees memory allocated to an <i>Imsls_f_nb_classifier</i> data structure.
<b>nb_classifier_write</b>	Writes a Naïve Bayes Classifier to an ASCII file for later retrieval using <i>imsls_f_nb_classifier_read</i> .
<b>nb_classifier_read</b>	Retrieves a Naïve Bayes Classifier from a file previously saved using <i>imsls_f_nb_classifier_write</i> .



### NEURAL NETWORK DATA STRUCTURES

FUNCTION	DESCRIPTION
<b>mlff_network_init</b>	Initializes a data structure for training a neural network.
<b>mlff_network</b>	Multilayered feedforward neural network.
<b>mlff_network_free</b>	Frees memory allocated for an <i>Imsls_f_NN_Network</i> data structure.
<b>mlff_network_write</b>	Writes a trained neural network to an ASCII file.
<b>mlff_network_read</b>	Retrieves a neural network from a file previously saved.
<b>mlff_initialize_weights</b>	Initializes weights for multilayered feedforward neural networks prior to network training using one of four user selected methods.

### FORECASTING NEURAL NETWORKS

FUNCTION	DESCRIPTION
<b>mlff_network_trainer</b>	Trains a multilayered feedforward neural network.
<b>mlff_network_forecast</b>	Calculates forecasts for trained multilayered feedforward neural networks.

### CLASSIFICATION NEURAL NETWORKS

FUNCTION	DESCRIPTION
<b>mlff_classification_trainer</b>	Trains a multilayered feedforward neural network for classification.

## CLASSIFICATION NEURAL NETWORKS (CONTINUED)

FUNCTION	DESCRIPTION
<b>mlff_pattern_classification</b>	Calculates classifications for trained multilayered feedforward neural networks.

## PREPROCESSING FILTERS

FUNCTION	DESCRIPTION
<b>scale_filter</b>	Scales or unscales continuous data prior to its use in neural network training, testing, or forecasting.
<b>time_series_filter</b>	Converts time series data to the format required for processing by a neural network.
<b>time_series_class_filter</b>	Converts time series data sorted within nominal classes in decreasing chronological order to a useful format for processing by a neural network.
<b>unsupervised_nominal_filter</b>	Converts nominal data into a series of binary encoded columns for input to a neural network. Optionally, it can also reverse the binary encoding, accepting a series of binary encoded columns and returning a single column of nominal classes.
<b>unsupervised_ordinal_filter</b>	Converts ordinal data into proportions. Optionally, it can also reverse encoding, accepting proportions and converting them into ordinal values.

## SELF ORGANIZING MAPS

FUNCTION	DESCRIPTION
<b>kohonenSOM_trainer</b>	Trains a Kohonen network.
<b>kohonenSOM_forecast</b>	Calculates forecasts using a trained Kohonen network.

SUPPORT VECTOR MACHINES	
FUNCTION	DESCRIPTION
<code>support_vector_trainer</code>	Trains a Support Vector Machines (SVM) classifier.
<code>support_vector_classification</code>	Classifies unknown patterns using a previously trained Support Vector Machines (SVM) model computed by <code>support_vector_trainer</code> .
<code>svm_classifier_free</code>	Frees memory allocated to an <code>Imsls_f_svm_model/</code> <code>Imsls_d_svm_model</code> data structure.

---

## CHAPTER 14: PRINTING FUNCTIONS

### PRINT

FUNCTION	DESCRIPTION
<code>write_matrix</code>	Prints a rectangular matrix (or vector) stored in contiguous memory locations.

### SET

FUNCTION	DESCRIPTION
<code>page</code>	Sets or retrieves the page width or length.
<code>write_options</code>	Sets or retrieves an option for printing a matrix.

## CHAPTER 15: UTILITIES

### SET OUPUT FILES

FUNCTION	DESCRIPTION
<code>output_file</code>	Sets the output file or the error message output file.
<code>version</code>	Returns integer information describing the version of the library, license number, operating system, and compiler.

### ERROR HANDLING

FUNCTION	DESCRIPTION
<code>error_options</code>	Sets various error handling options.
<code>error_type</code>	Gets the type corresponding to the error message from the last function called.
<code>error_message</code>	Gets the text of the error message from the last function called.
<code>error_code</code>	Returns the code corresponding to the error message from the last function called.
<code>initialize_error_handler</code>	Initializes the IMSL C Stat Library error handling system.
<code>set_user_fcn_return_flag</code>	Indicates a condition has occurred in a user-supplied function necessitating a return to the calling function.

C RUNTIME LIBRARY	
FUNCTION	DESCRIPTION
<b>free</b>	Frees memory returned from an IMSL C Stat Library function.
<b>fopen</b>	Opens a file using the C runtime library used by the IMSL C Stat Library.
<b>fclose</b>	Closes a file opened by <code>imsls_fopen</code> .
<b>ascii_read</b>	Reads freely-formatted ASCII files.

OPENMP	
FUNCTION	DESCRIPTION
<b>omp_options</b>	Sets various OpenMP options.

CONSTANTS	
FUNCTION	DESCRIPTION
<b>machine (integer)</b>	Returns integer information describing the computer's arithmetic.
<b>machine (float)</b>	Returns information describing the computer's floating-point arithmetic.
<b>data_sets</b>	Retrieves a commonly analyzed data set.

MATHEMATICAL SUPPORT	
FUNCTION	DESCRIPTION
<b>mat_mul_rect</b>	Computes the transpose of a matrix, a matrix-vector product, a matrix-matrix product, a bilinear form, or any triple product.
<b>permute_vector</b>	Rearranges the elements of a vector as specified by a permutation.
<b>permute_matrix</b>	Permutates the rows or columns of a matrix.
<b>impute_missing</b>	Locates and optionally replaces dependent variable missing values with nearest neighbor estimates.
<b>binomial_coefficient</b>	Evaluates the binomial coefficient.
<b>beta</b>	Evaluates the real regularized incomplete beta function.
<b>beta_incomplete</b>	Evaluates the real regularized incomplete beta function.
<b>log_beta</b>	Evaluates the logarithm of the real beta function $\ln \beta(x, y)$ .
<b>gamma</b>	Evaluates the real gamma function.
<b>gamma_incomplete</b>	Evaluates the incomplete gamma function $\gamma(a, x)$ .
<b>log_gamma</b>	Evaluates the logarithm of the absolute value of the gamma function $\log  \Gamma(x) $ .
<b>ctime</b>	Returns the number of CPU seconds used.

GPU SUPPORT	
FUNCTION	DESCRIPTION
<code>cuda_get</code>	Gets the threshold used by the specified function to determine if the <i>NVIDIA® CUDA™ Toolkit</i> algorithm will be used.
<code>cuda_set</code>	Sets the threshold used by the specified function to determine if the <i>NVIDIA® CUDA™ Toolkit</i> algorithm will be used.
<code>cuda_free</code>	Releases NVIDIA memory allocated by the IMSL C Numerical Library.