IMSL® C Numerical Library

Function Catalog

Version 7.0

ROGUE WAVE®
SOFTWARE

CORPORATE HEADQUARTERS
Rogue Wave Software
5500 Flatiron Parkway Suite 200
Boulder, CO 80301 USA
www.roguewave.com

IMSL Libraries Contact Information
Contact Support

Products Overview
Products Overview

© 1970-2010 Rogue Wave Software. The Rogue Wave Software name and logo, IMSL and PV-WAVE are registered trademarks of Rogue Wave Software, Inc. or its subsidiaries in the US and other countries. JMSL, JWAVE, TS-WAVE, and PyIMSL are trademarks of Rogue Wave Software, Inc. or its subsidiaries. All other company, product or brand names are the property of their respective owners.

IMPORTANT NOTICE: Information contained in this documentation is subject to change without notice. Use of this document is subject to the terms and conditions of a Rogue Wave Software License Agreement, including, without limitation, the Limited Warranty and Limitation of Liability. If you do not accept the terms of the license agreement, you may not use this documentation and should promptly return the product for a full refund. This documentation may not be copied or distributed in any form without the express written consent of Rogue Wave.

IMSL® C, C#, Java™ and Fortran Application Development Tools

IMSL C Numerical Library Version 7.0 3

- Overview 3
- Mathematical Functionality 6
- Statistical Functionality 7
- IMSL Library - Also Available for Java™, C# and Fortran 8

IMSL C/Math/Library 9

- CHAPTER 1: Linear Systems 9
- CHAPTER 2: Eigensystem Analysis 10
- CHAPTER 3: Interpolation and Approximation 10
- CHAPTER 4: Quadrature 12
- CHAPTER 5: Differential Equations 13
- CHAPTER 6: Transforms 14
- CHAPTER 7: Nonlinear Equations 15
- CHAPTER 8: Optimization 15
- CHAPTER 9: Special Functions 16
- CHAPTER 10: Statistics and Random Number Generation 21
- CHAPTER 11: Printing Functions 22
- CHAPTER 12: Utilities 23

IMSL C/Stat/Library 27

- CHAPTER 1: Basic Statistics 27
- CHAPTER 2: Regression 27
- CHAPTER 3: Correlation and Covariance 28
- CHAPTER 4: Analysis of Variance and Designed Experiments 29
- CHAPTER 5: Categorical and Discrete Data Analysis 30
- CHAPTER 6: Nonparametric Statistics 30
- CHAPTER 7: Tests of Goodness of Fit 31
- CHAPTER 8: Time Series and Forecasting 31
- CHAPTER 9: Multivariate Analysis 32
- CHAPTER 10: Survival and Reliability Analysis 33
- CHAPTER 11: Probability Distribution Functions and Inverses 33
- CHAPTER 12: Random Number Generation 35
- CHAPTER 13: Data Mining 38
- CHAPTER 14: Printing Functions 40
- CHAPTER 15: Utilities 40

IMSL® C NUMERICAL LIBRARY VERSION 7.0

For C and C++ programmers, providing the broadest coverage available of numerical functions written in native C.

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

COST-EFFECTIVENESS AND VALUE

The IMSL C Numerical Library significantly shortens program development time and promotes standardization. Variable argument lists have been implemented to simplify calling sequences. The IMSL C Library saves time in source code development and saves thousands of dollars in the design, development, documentation, testing and maintenance of applications.

SMP ENABLED

The IMSL C Library is 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 vendor libraries leverage SMP capabilities on a variety of systems.

USER-FRIENDLY NOMENCLATURE

The IMSL C Library uses descriptive, explanatory function names for intuitive programming. Where appropriate, consistent variable names are used to:

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

ERROR HANDLING

Diagnostic error messages are clear and informative – designed not only to convey the error condition but also to suggest corrective action if appropriate.

These configurable error-handling features:

- Make it faster and easier to debug programs
- Provide for more productive programming and confidence that the algorithms are functioning properly in an application

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. The concise set of required arguments contains only information necessary for usage. Optional arguments provide added functionality and power to each function.

This flexibility:

- Reduces unnecessary code
- Enables the user to tailor each function call to specific program requirements.

WIDE COMPATIBILITY AND UNIFORM OPERATION

With the IMSL Library, it is easy to build applications that are portable across multiple platforms. The IMSL C Library is available for a wide selection of UNIX/Linux and Windows computing environments.

Visual Numerics' commitment to regular feature and enhancement updates:

- Ensures that software will perform to the highest standards
- Provides for portable applications
- Assures that Visual Numerics will keep pace with the latest hardware and software innovations

SHARED LIBRARY TECHNOLOGY

The IMSL C Library is designed to take advantage of shared libraries technology.

This technology:

- Allows more than one user to share code in the library thus minimizing disk space usage
- Provides shorter link time
- Minimizes the size of executable object modules

THREAD SAFETY

The IMSL C Library is thread safe. Thread safety allows it 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 Library is comprehensive, clearly written and standardized. The documentation, in hardcopy or online formats:

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

UNMATCHED PRODUCT SUPPORT

Behind every Visual Numerics' license is a team of professionals ready to provide expert answers to questions about the IMSL Numerical Libraries. Product support options include product maintenance, ensuring the value and performance of IMSL Library software.

Product support:

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

CONSULTING SERVICES

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

ACCESS FROM PYTHON

IMSL C Library functions can be accessed from Python through PyIMSL™, a collection of Python wrappers to the mathematical and statistical algorithms in IMSL C Numerical Library. PyIMSL is a complimentary add-on to IMSL C Library. PyIMSL is unique in that developers get consistency between the mathematical and statistical algorithms used for prototyping and those used for production. With PyIMSL, developers in Python can access IMSL C Library functions for rapid prototyping and then directly deploy the Python application into production, or if they choose to rewrite the application in C/C++, use the same mathematical and statistical algorithms in both development environments. For more information, please visit www.vni.com/pyimsl.

Mathematical Functionality

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 10 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.

Eigensystem 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 and Gauss quadrature.

Differential Equations, using Adams-Gear and Runge-Kutta methods for stiff and nonstiff ordinary differential equations and support for partial differential equations.

Transforms, including real and complex one- and two-dimensional fast Fourier transforms, as well as convolutions and 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, and linearly and nonlinearly constrained minimizations.

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

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

Utilities, including CPU time used, error handling and machine, mathematical, physical constants, retrieval of machine constants, changing error handling defaults, and performing matrix-matrix multiplication.

Statistical Functionality

The statistical functionality is organized into 13 sections. These capabilities range from analysis of variance to random number generation.

Basic Statistics, including univariate summary statistics, nonparametric tests, such as sign and Wilcoxon rank sum, and goodness-of-fit tests, such as chi-squared and Shapiro-Wilks' tests.

Regression, including stepwise regression, all best regression, multiple linear regression models, polynomial models and nonlinear models.

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, including Yates' method for estimating missing observations in designed experiments, analysis of hierarchical data, analysis of standard factorial experiments, randomized complete block designs, latin-square, lattice, split-plot, strip-plot, split-split plot and strip-split plot experiments, and standard tests for multiple comparisons of treatment means 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 and analysis of categorical data using general linear models, including logistic regression.

Nonparametric Statistics, including sign tests, Wilcoxon rank sum tests and Cochran's Q test for related observations.

Tests of Goodness-of-Fit, including chi-squared goodness-of-fit tests, Kolmogorov/Smirnov tests and tests for normality.

Time Series Analysis and Forecasting, including analysis and forecasting of time series using a nonseasonal ARMA model, GARCH (Generalized Autoregressive Conditional Heteroskedasticity), Kalman filtering, 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 components, 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 binomial, hypergeometric, bivariate normal, gamma and many more.

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.

Data Mining, including feed forward neural networks, plus neural network data pre- and post-processing algorithms, are particularly well suited to developing predictive models in noisy or challenging data situations.

IMSL Library – Also Available for JAVA™, C# and Fortran

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 data analysis and visualization technology for the Java platform and a fast, scalable framework for tailored analytical applications. In addition, the JMSL Library includes neural network technology that adds to the broad selection of existing data mining, modeling and prediction technologies available across the IMSL Family of products. These neural network forecasting classes have tremendous potential for businesses by offering the ability to build predictive models using historical data and training the network to optimize the model over time as more information is obtained.

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

Written in 100% C#, the IMSL C# Library for Microsoft .NET offers .NET developers seamless accessibility to analytics capabilities with the highest degree of programming productivity and ease of use with Visual Studio™.

As the only numerical library of its kind to offer industry standard numerical analysis and charting for .NET languages, the IMSL C# Library can be referenced from any .NET language including C#, F# and Visual Basic™.NET. It provides unprecedented analytic capabilities and the most comprehensive and accessible mathematical, statistical and finance algorithms for the .NET Framework. With the IMSL C# Library, Visual Numerics has brought all of the benefits of the .NET platform to a new level by adding robust analytics to its broad set of capabilities.

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.

CHAPTER 1: LINEAR SYSTEMS

LINEAR EQUATIONS WITH FULL MATRICES:	
lin_sol_gen	Solves a real general system of linear equations $Ax = b$.
lin_sol_gen (complex)	Solves a complex general system of linear equations $Ax = b$.
lin_sol_posdef	Solves a real symmetric positive definite system of linear equations $Ax = b$.
lin_sol_posdef (complex)	Solves a complex Hermitian positive definite system of linear equations $Ax = b$.
LINEAR EQUATIONS WITH BAND MATRICES:	
lin_sol_gen_band	Solves a real general band system of linear equations $Ax = b$.
lin_sol_gen_band (complex)	Solves a complex general band system of linear equations $Ax = b$.
lin_sol_posdef_band	Solves a real symmetric positive definite system of linear equations $Ax = b$ in band symmetric storage mode.
lin_sol_posdef_band (complex)	Solves a complex Hermitian positive definite system of linear equations $Ax = b$ in band symmetric storage mode.
LINEAR EQUATIONS WITH GENERAL SPARSE MATRICES:	
lin_sol_gen_coordinate	Solves a sparse system of linear equations $Ax = b$.
lin_sol_gen_coordinate (complex)	Solves a sparse system of linear equations $Ax = b$, with sparse complex coefficient matrix A.
lin_sol_posdef_coordinate	Solves a sparse real symmetric positive definite system of linear equations $Ax = b$.
lin_sol_posdef_coordinate (complex)	Solves a sparse Hermitian positive definite system of linear equations $Ax = b$.
ITERATIVE METHODS:	
lin_sol_gen_min_residual	Solves a linear system $Ax = b$ using the restarted generalized minimum residual (GMRES) method.

ITERATIVE METHODS: (CON'T)

lin_sol_def_cg	Solves a real symmetric definite linear system using a conjugate gradient method.

LINEAR LEAST-SQUARES WITH FULL MATRICES:

lin_least_squares_gen	Solves a linear least-squares problem $Ax = b$.
lin_lsq_lin_constraints	Solves a linear least squares problem with linear constraints.
lin_svd_gen	Computes the SVD, $A = USV^T$, of a real rectangular matrix A.
lin_svd_gen (complex)	Computes the SVD, $A = USV^H$, of a complex rectangular matrix A.
lin_sol_nonnegdef	Solves a real symmetric nonnegative definite system of linear equations $Ax = b$.

CHAPTER 2: EIGENSYSTEM ANALYSIS

LINEAR EIGENSYSTEM PROBLEMS:

eig_gen	Computes the eigenexpansion of a real matrix A.
eig_gen (complex)	Computes the eigenexpansion of a complex matrix A.
eig_sym	Computes the eigenexpansion of a real symmetric matrix A.
eig_herm (complex)	Computes the eigenexpansion of a complex Hermitian matrix A.

GENERALIZED EIGENSYSTEM PROBLEMS:

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

CHAPTER 3: INTERPOLATION AND APPROXIMATION

CUBIC SPLINE INTERPOLATION:

cub_spline_interp_e_cnd	Computes a cubic spline interpolant, specifying various endpoint conditions.

CUBIC SPLINE INTERPOLATION: (CON'T)

cub_spline_interp_shape	Computes a shape-preserving cubic spline.
cub_spline_tcb	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:

cub_spline_value	Computes the value of a cubic spline or the value of one of its derivatives.
cub_spline_integral	Computes the integral of a cubic spline.

SPLINE INTERPOLATION:

spline_interp	Computes a spline interpolant.
spline_knots	Computes the knots for a spline interpolant.
spline_2d_interp	Computes a two-dimensional, tensor-product spline interpolant from two-dimensional, tensor-product data.

SPLINE EVALUATION AND INTEGRATION:

spline_value	Computes the value of a spline or the value of one of its derivatives.
spline_integral	Computes the integral of a spline.
spline_2d_value	Computes the value of a tensor-product spline or the value of one of its partial derivatives.
spline_2d_integral	Evaluates the integral of a tensor-product spline on a rectangular domain.

LEAST-SQUARES APPROXIMATION AND SMOOTHING:

user_fcn_least_squares	Computes a least-squares fit using user-supplied functions.
spline_least_squares	Computes a least-squares spline approximation.
spline_2d_least_squares	Computes a two-dimensional, tensor-product spline approximant using least squares.
cub_spline_smooth	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.
spline_lsq_constrained	Computes a least-squares constrained spline approximation.
smooth_1d_data	Smooth one-dimensional data by error detection.

SCATTERED DATA INTERPOLATION:	
scattered_2d_interp	Computes a smooth bivariate interpolant to scattered data that is locally a quintic polynomial in two variables.

SCATTERED DATA LEAST SQUARES:	
radial_scattered_fit	Computes an approximation to scattered data in R^n for $n \leq 1$ using radial basis functions.
radial_evaluate	Evaluates a radial basis fit.

CHAPTER 4: QUADRATURE

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

MULTIVARIATE QUADRATURE:	
int_fcn_2d	Computes a two-dimensional iterated integral.
int_fcn_hyper_rect	Integrates a function on a hyper-rectangle $\int_{a_0}^{b_0} \ldots \int_{a_{n-1}}^{b_{n-1}} f(x_0, \ldots, x_{n-1})\, dx_{n-1} \ldots dx_0$.
int_fcn_qmc	Integrates a function on a hyper-rectangle using a quasi-Monte Carlo method.

GAUSS QUADRATURE:	
gauss_quad_rule	Computes a Gauss, Gauss-Radau, or Gauss-Lobatto quadrature rule with various classical weight functions.

DIFFERENTIATION:	
fcn_derivative	Computes the first, second or third derivative of a user-supplied function.

CHAPTER 5: DIFFERENTIAL EQUATIONS

RUNGE-KUTTA METHOD:	
ode_runge_kutta	Solves an initial-value problem for ordinary differential equations using the Runge-Kutta-Verner fifth-order and sixth-order method.
ADAM'S OR GEAR'S METHOD:	
ode_adams_gear	Solves a stiff initial-value problem for ordinary differential equations using the Adams-Gear methods.
PETZOLD-GEAR METHOD:	
dea_petzold_gear	Solves a first order differential-algebraic system of equations, $g(t, y, y^*) = 0$, using the Petzold-Gear BDF method.
PARTIAL DIFFERENTIAL EQUATIONS:	
pde_1d_mg	Solves a system of one-dimensional time-dependent partial differential equations using a moving-grid interface.
feynman_kac	Solves a generalized Feynman-Kac equation on a finite interval using Hermite quintic splines.
feynman_kac_evaluate	Computes the value of a Hermite quintic spline or the value of one of its derivatives.
METHOD OF LINES:	
pde_method_of_lines	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.
BOUNDARY VALUE PROBLEM:	
bvp_finite_difference	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.

FAST POISSON SOLVER:	
fast_poisson_2d	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 FFTS:	
fft_real	Computes the real discrete Fourier transform of a real sequence.
fft_real_init	Computes the parameters for imsl_f_fft_real.
COMPLEX EXPONENTIAL FFTS:	
fft_complex	Computes the complex discrete Fourier transform of a complex sequence.
fft_complex_init	Computes the parameters for imsl_c_fft_complex.
REAL SINE AND COSINE FFTS:	
fft_cosine	Computes the discrete Fourier cosine transformation of an even sequence.
fft_cosine_init	Computes the parameters needed for imsl_f_fft_cosine.
fft_sine	Computes the discrete Fourier sine transformation of an odd sequence.
fft_sine_init	Computes the parameters needed for imsl_f_fft_sine.
TWO-DIMENSIONAL FFTS:	
fft_2d_complex	Computes the complex discrete two-dimensional Fourier transform of a complex two-dimensional array.
CONVOLUTION AND CORRELATION:	
convolution	Computes the convolution, and optionally, the correlation of two real vectors.
convolution (complex)	Computes the convolution, and optionally, the correlation of two complex vectors.
LAPLACE TRANSFORM:	
inverse_laplace	Computes the inverse Laplace transform of a complex function.

CHAPTER 7: NONLINEAR EQUATIONS

ZEROS OF A POLYNOMIAL:	
zeros_poly	Finds the zeros of a polynomial with real coefficients using the Jenkins-Traub three-stage algorithm.
zeros_poly (complex)	Finds the zeros of a polynomial with complex coefficients using the Jenkins-Traub three-stage algorithm.
ZEROS OF A FUNCTION:	
zeros_function	Finds the real zeros of a real, continuous, univariate function.
ROOT OF A SYSTEM OF EQUATIONS:	
zeros_sys_eqn	Solves a system of n nonlinear equations $f(x) = 0$ using a modified Powell hybrid algorithm.

CHAPTER 8: OPTIMIZATION

UNCONSTRAINED MINIMIZATION:	
min_uncon	Finds the minimum point of a smooth function $f(x)$ of a single variable using only function evaluations.
min_uncon_deriv	Finds the minimum point of a smooth function $f(x)$ of a single variable using both function and first derivative evaluations.
min_uncon_golden	Finds the minimum point of a nonsmooth function of a single variable using the golden section search method.
min_uncon_multivar	Minimizes a function $f(x)$ of n variables using a quasi-Newton method.
nonlin_least_squares	Solves a nonlinear least-squares problem using a modified Levenberg-Marquardt algorithm.
LINEARLY CONSTRAINED MINIMIZATION:	
read_mps	Reads an MPS file containing a linear programming problem or a quadratic programming problem.
linear_programming	Solves a linear programming problem.
lin_prog	Solves a linear programming problem using the revised simplex algorithm.
quadratic_prog	Solves a quadratic programming problem subject to linear equality or inequality constraints.

OPTIMIZATION: (CON'T)

min_con_gen_lin	Minimizes a general objective function subject to linear equality/inequality constraints.
bounded_least_squares	Solves a nonlinear least-squares problem subject to bounds on the variables using a modified Levenberg-Marquardt algorithm.

NONLINEARLY CONSTRAINED MINIMIZATION:

constrained_nlp	Solves a general nonlinear programming problem using a sequential equality constrained quadratic programming method.

CHAPTER 9: SPECIAL FUNCTIONS

ERROR AND GAMMA FUNCTIONS:

erf	Evaluates the real error function erf*(x).*
erfc	Evaluates the real complementary error function erfc*(x).*
erf_inverse	Evaluates the real inverse error function erf^{-1} *(x).*
erfce	Evaluates the exponentially scaled complementary error function.
erfe	Evaluates a scaled function related to erfc*(z).*
erfc_inverse	Evaluates the real inverse complementary error function erfc^{-1} *(x).*
beta	Evaluates the real beta function β *(x, y).*
log_beta	Evaluates the logarithm of the real beta function ln β *(x, y).*
beta_incomplete	Evaluates the real incomplete beta function $I_x = \beta_x(a,b)/\beta(a,b)$.
gamma	Evaluates the real gamma function $\Gamma(x)$.
log_gamma	Evaluates the logarithm of the absolute value of the gamma function log $\lvert\Gamma(x)\rvert$.
gamma_incomplete	Evaluates the incomplete gamma function $\gamma(a, x)$.

BESSEL FUNCTIONS:

bessel_J0	Evaluates the real Bessel function of the first kind of order zero $J_0(x)$.
bessel_J1	Evaluates the real Bessel function of the first kind of order one $J_1(x)$.

BESSEL FUNCTIONS: (CON'T)

bessel_Jx	Evaluates a sequence of Bessel functions of the first kind with real order and complex arguments.
bessel_Y0	Evaluates the real Bessel function of the second kind of order zero $Y_0(x)$.
bessel_Y1	Evaluates the real Bessel function of the second kind of order one $Y_1(x)$.
bessel_Yx	Evaluates a sequence of Bessel functions of the second kind with real order and complex arguments.
bessel_I0	Evaluates the real modified Bessel function of the first kind of order zero $I_0(x)$.
bessel_exp_I0	Evaluates the exponentially scaled modified Bessel function of the first kind of order zero.
bessel_I1	Evaluates the real modified Bessel function of the first kind of order one $I_1(x)$.
bessel_exp_I1	Evaluates the exponentially scaled modified Bessel function of the first kind of order one.
bessel_Ix	Evaluates a sequence of modified Bessel functions of the first kind with real order and complex arguments.
bessel_K0	Evaluates the real modified Bessel function of the second kind of order zero $K_0(x)$.
bessel_exp_K0	Evaluates the exponentially scaled modified Bessel function of the second kind of order zero.
bessel_K1	Evaluates the real modified Bessel function of the second kind of order one $K_1(x)$.
bessel_exp_K1	Evaluates the exponentially scaled modified Bessel function of the second kind of order one.
bessel_Kx	Evaluates a sequence of modified Bessel functions of the second kind with real order and complex arguments.

ELLIPTIC INTEGRALS:

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

FRESNEL INTEGRALS:	
fresnel_integral_C	Evaluates the cosine Fresnel integral.
fresnel_integral_S	Evaluates the sine Fresnel integral.
AIRY FUNCTIONS:	
airy_Ai	Evaluates the Airy function.
airy_Bi	Evaluates the Airy function of the second kind.
airy_Ai_derivative	Evaluates the derivative of the Airy function.
airy_Bi_derivative	Evaluates the derivative of the Airy function of the second kind.
KELVIN FUNCTIONS:	
kelvin_ber0	Evaluates the Kelvin function of the first kind, ber, of order zero.
kelvin_bei0	Evaluates the Kelvin function of the first kind, bei, of order zero.
kelvin_ker0	Evaluates the Kelvin function of the second kind, ker, of order zero.
kelvin_kei0	Evaluates the Kelvin function of the second kind, kei, of order zero.
kelvin_ber0_derivative	Evaluates the derivative of the Kelvin function of the first kind, ber, of order zero.
kelvin_bei0_derivative	Evaluates the derivative of the Kelvin function of the first kind, bei, of order zero.
kelvin_ker0_derivative	Evaluates the derivative of the Kelvin function of the second kind, ker, of order zero.
kelvin_kei0_derivative	Evaluates the derivative of the Kelvin function of the second kind, kei, of order zero.
STATISTICAL FUNCTIONS:	
normal_cdf	Evaluates the standard normal (Gaussian) distribution function.
normal_inverse_cdf	Evaluates the inverse of the standard normal (Gaussian) distribution function.
chi_squared_cdf	Evaluates the chi-squared distribution function.
chi_squared_inverse_cdf	Evaluates the inverse of the chi-squared distribution function.
F_cdf	Evaluates the F distribution function.
F_inverse_cdf	Evaluates the inverse of the F distribution function.

STATISTICAL FUNCTIONS: (CON'T)

t_cdf	Evaluates the Student's t distribution function.
t_inverse_cdf	Evaluates the inverse of the Student's t distribution function.
gamma_cdf	Evaluates the gamma distribution function.
binomial_cdf	Evaluates the binomial distribution function.
hypergeometric_cdf	Evaluates the hypergeometric distribution function.
poisson_cdf	Evaluates the Poisson distribution function.
beta_cdf	Evaluates the beta distribution function.
beta_inverse_cdf	Evaluates the inverse of the beta distribution function.
bivariate_normal_cdf	Evaluates the bivariate normal distribution function.

FINANCIAL FUNCTIONS:

cumulative_interest	Evaluates the cumulative interest paid between two periods.
cumulative_principal	Evaluates the cumulative principal paid between two periods.
depreciation_db	Evaluates the depreciation of an asset. (Fixed-declining balance method.)
depreciation_ddb	Evaluates the depreciation of an asset. (Double-declining balance method.)
depreciation_sln	Evaluates the depreciation of an asset. (Straight-line method.)
depreciation_syd	Evaluates the depreciation of an asset. (Sum-of-years digits method.)
depreciation_vdb	Evaluates the depreciation of an asset for any given period, including partial periods. (Variable-declining balance method.)
dollar_decimal	Converts a fractional price to a decimal price.
dollar_fraction	Converts a decimal price to a fractional price.
effective_rate	Evaluates the effective annual interest rate.
future_value	Evaluates an investment's future value.
future_value_schedule	Evaluates the future value of an initial principal taking into consideration a schedule of compound interest rates.
interest_payment	Evaluates the interest payment for an investment for a given period.

FINANCIAL FUNCTIONS: (CON'T)

interest_rate_annuity	Evaluates an annuity's interest rate per period.
internal_rate_of_return	Evaluates the internal rate of return for a schedule of cash flows.
internal_rate_schedule	Evaluates the internal rate of return for a schedule of cash flows. It is not necessary that the cash flows be periodic.
modified_internal_rate	Evaluates the modified internal rate of return for a schedule of periodic cash flows.
net_present_value	Evaluates an investment's net present value. The calculation is based on a schedule of periodic cash flows and a discount rate.
nominal_rate	Evaluates the nominal annual interest rate.
number_of_periods	Evaluates the number of periods for an investment for which periodic and constant payments are made and the interest rate is constant.
payment	Evaluates the periodic payment for an investment.
present_value	Evaluates the net present value of a stream of equal periodic cash flows, which are subject to a given discount rate.
present_value_schedule	Evaluates the present value for a schedule of cash flows. It is not necessary that the cash flows be periodic.
principal_payment	Evaluates the payment on the principal for a specified period.

BOND FUNCTIONS:

accr_interest_maturity	Evaluates the interest that has accrued on a security, which pays interest at maturity.
accr_interest_periodic	Evaluates the interest that has accrued on a security, which pays interest periodically.
bond_equivalent_yield	Evaluates a Treasury bill's bond-equivalent yield.
convexity	Evaluates the convexity for a security.
coupon_days	Evaluates the number of days in the coupon period containing the settlement date.
coupon_number	Evaluates the number of coupons payable between the settlement date and the maturity date.
days_before_settlement	Evaluates the number of days starting with the beginning of the coupon period and ending with the settlement date.
days_to_next_coupon	Evaluates the number of days starting with the settlement date and ending with the next coupon date.
depreciation_amordegrc	Evaluates the depreciation for each accounting period. During the evaluation of the function a depreciation coefficient based on the asset life is applied.

BOND FUNCTIONS: (CON'T)

depreciation_amorlinc	Evaluates the depreciation for each accounting period. This function is similar to *depreciation_amordegrc*, except that *depreciation_amordegrc* has a depreciation coefficient that is applied during the evaluation that is based on the asset life.
discount_price	Evaluates a discounted security's price per $100 face value.
discount_rate	Evaluates a security's discount rate.
discount_yield	Evaluates a discounted security's annual yield.
duration	Evaluates a security's annual duration where the security has periodic interest payments.
interest_rate_security	Evaluates a fully invested security's interest rate.
modified_duration	Evaluates a security's modified Macauley duration assuming a par value of $100.
next_coupon_date	Evaluates the first coupon date that follows the settlement date.
previous_coupon_date	Evaluates the coupon date that immediately precedes the settlement date.
price	Evaluates a security's price per $100 face value. The security pays periodic interest.
price_maturity	Evaluates a security's price per $100 face value. The security pays interest at maturity.
received_maturity	Evaluates the amount one receives when a fully invested security reaches the maturity date.
treasury_bill_price	Evaluates a Treasury bill's price per $100 face value.
treasury_bill_yield	Evaluates a Treasury bill's yield.
year_fraction	Evaluates the fraction of a year represented by the number of whole days between two dates.
yield_maturity	Evaluates a security's annual yield. The security pays interest at maturity.
yield_periodic	Evaluates a security's yield. The security pays periodic interest.

CHAPTER 10: STATISTICS AND RANDOM NUMBER GENERATION

STATISTICS:

simple_statistics	Computes basic univariate statistics.
table_oneway	Tallies observations into a one-way frequency table.
chi_squared_test	Performs a chi-squared goodness-of-fit test.

STATISTICS: (CON'T)

covariances	Computes the sample variance-covariance or correlation matrix.
regression	Fits a multiple linear regression model using least squares.
poly_regression	Performs a polynomial least-squares regression.
ranks	Computes the ranks, normal scores, or exponential scores for a vector of observations.

RANDOM NUMBERS:

random_seed_get	Retrieves the current value of the seed used in the IMSL random number generators.
random_seed_set	Initializes a random seed for use in the IMSL random number generators.
random_option	Selects the uniform (0, 1) multiplicative congruential pseudorandom number generator.
random_uniform	Generates pseudorandom numbers from a uniform (0, 1) distribution.
random_normal	Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.
random_poisson	Generates pseudorandom numbers from a Poisson distribution.
random_gamma	Generates pseudorandom numbers from a standard gamma distribution.
random_beta	Generates pseudorandom numbers from a beta distribution.
random_exponential	Generates pseudorandom numbers from a standard exponential distribution.
faure_next_point	Computes a shuffled Faure sequence.

CHAPTER 11: PRINTING FUNCTIONS

PRINT:

write_matrix	Prints a rectangular matrix (or vector) stored in contiguous memory locations.

SET:

page	Sets or retrieves the page width or length.
write_options	Sets or retrieves an option for printing a matrix.

CHAPTER 12: UTILITIES

SET OUTPUT FILES:	
output_file	Sets the output file or the error message output file.
version	Returns integer information describing the version of the library, license number, operating system, and compiler.
TIME AND DATE:	
ctime	Returns the number of CPU seconds used.
days_to_date	Gives the date corresponding to the number of days since January 1, 1900.
date_to_days	Computes the number of days from January 1, 1900, to the given date.
ERROR HANDLING:	
error_code	Gets the code corresponding to the error message from the last function called.
error_message	Gets the text of the error message from the last function called.
error_options	Sets various error handling options.
error_type	Gets the type corresponding to the error message from the last function called.
C RUNTIME LIBRARY:	
initialize	Initializes the IMSL C Math Library.
free	Frees memory returned from an IMSL C Math Library function.
fopen	Opens a file using the C runtime library used by the IMSL C Math Library.
fclose	Closes a file opened by imsl_fopen.
OpenMP:	
omp_options	Sets various OpenMP options.
CONSTANTS:	
constant	Returns the value of various mathematical and physical constants.
machine (float)	Returns information describing the computer's floating-point arithmetic.
machine (integer)	Returns integer information describing the computer's arithmetic.

SORTING:	
sort	Sorts a vector by algebraic value. Optionally, a vector can be sorted by absolute value, and a sort permutation can be returned.
sort (integer)	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:	
vector_norm	Computes various norms of a vector or the difference of two vectors.
LINEAR ALGEBRA SUPPORT:	
mat_mul_rect	Computes the transpose of a matrix, a matrix-vector product, a matrix-matrix product, the bilinear form, or any triple product.
mat_mul_rect (complex)	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.
mat_mul_rect_band	Computes the transpose of a matrix, a matrix-vector product, or a matrix-matrix product, all matrices stored in band form.
mat_mul_rect_band (complex)	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.
mat_mul_rect_coordinate	Computes the transpose of a matrix, a matrix-vector product, or a matrix-matrix product, all matrices stored in sparse coordinate form.
mat_mul_rect_coordinate (complex)	Computes the transpose of a matrix, a matrix-vector product or a matrix-matrix product, all matrices stored in sparse coordinate form.
mat_add_band	Adds two band matrices, both in band storage mode, $C \leftarrow \alpha A + \beta B$.
mat_add_band (complex)	Adds two band complex matrices, both in band storage mode, $C \leftarrow \alpha A + \beta B$.
mat_add_coordinate	Performs element-wise addition of two real matrices stored in coordinate format, $C \leftarrow \alpha A + \beta B$.
mat_add_coordinate (complex)	Performs element-wise addition on two complex matrices stored in coordinate format, $C \leftarrow \alpha A + \beta B$.
matrix_norm	Computes various norms of a rectangular matrix.
matrix_norm_band	Computes various norms of a matrix stored in band storage mode.
matrix_norm_coordinate	Computes various norms of a matrix stored in coordinate format.
generate_test_band	Generates test matrices of class $E(n, c)$.

BOND FUNCTIONS: (CON'T)

generate_test_band (complex)	Generates test matrices of class $E_c(n, c)$.
generate_test_coordinate	Generates test matrices of class $D(n, c)$ and $E(n, c)$.
generate_test_coordinate (complex)	Generates test matrices of class $D(n, c)$ and $E(n, c)$.

NUMERIC UTILITIES

c_neg	Changes the sign of a complex number.
c_add	Adds two complex numbers.
c_sub	Subtracts a complex number from a complex number.
c_mul	Multiplies two complex numbers.
c_div	Divides a complex number by a complex number.
c_eq	Tests for equality of two complex numbers.
cz_convert	Truncates a double precision complex number to a single precision complex number.
zc_convert	Increases precision of a single precision complex number to a double precision complex number.
cf_convert	Makes a complex number from an ordered pair.
c_conjg	Conjugates a complex number.
c_abs	Computes a magnitude of a complex number.
c_arg	Computes an angle of a complex number.
c_sqrt	Computes a square root of a complex number.
c_cos	Computes a trigonometric cosine of a complex number.
c_sin	Computes a trigonometric sine of a complex number.
c_exp	Computes an exponential function of a complex number.
c_log	Computes a natural logarithm of a complex number.
cf_power	Computes a complex number raised to a real power.
cc_power	Computes a complex number raised to a complex power.

NUMERIC UTILITIES: (CON'T)	
fi_power	Computes a real number raised to an integral power.
ii_power	Computes an integer raised to an integral power.

CHAPTER 1: BASIC STATISTICS

SIMPLE SUMMARY STATISTICS:	
simple_statistics	Computes basic univariate statistics.
empirical_quantiles	Computes empirical quantiles.
normal_one_sample	Computes statistics for mean and variance inferences using a sample from a normal population.
normal_two_sample	Computes statistics for mean and variance inferences using samples from two normal populations.
TABULATE, SORT, RANK:	
table_oneway	Tallies observations into a one-way frequency table.
table_twoway	Tallies observations into a two-way frequency table.
sort-data	Sorts observations by specified keys, with option to tally cases into a multi-way frequency table.
ranks	Computes the ranks, normal scores, or exponential scores for a vector of observations.

CHAPTER 2: REGRESSION

MULTIVARIATE LINEAR REGRESSION—MODEL FITTING:	
regressors_for_glm	Generates regressors for a general linear model.
regression	Fits a multiple linear regression model using least squares.
MULTIVARIATE LINEAR REGRESSION—STATISTICAL INFERENCE AND DIAGNOSTICS:	
regression_summary	Produces summary statistics for a regression model given the information from the fit.
regression_prediction	Computes predicted values, confidence intervals, and diagnostics after fitting a regression model.

MULTIVARIATE LINEAR REGRESSION—STATISTICAL INFERENCE AND DIAGNOSTICS: (CON'T)

hypothesis_partial	Constructs a completely testable hypothesis.
hypothesis_scph	Sums of cross products for a multivariate hypothesis.
hypothesis_test	Tests for the multivariate linear hypothesis.
VARIABLE SELECTION:	
regression_selection	Selects the best multiple linear regression models.
regression_stepwise	Builds multiple linear regression models using forward selection, backward selection or stepwise selection.
POLYNOMIAL AND NONLINEAR REGRESSION:	
poly_regression	Performs a polynomial least-squares regression.
poly_prediction	Computes predicted values, confidence intervals, and diagnostics after fitting a polynomial regression model.
nonlinear_regression	Fits a nonlinear regression model.
nonlinear_optimization	Fits a nonlinear regression model using Powell's algorithm.
ALTERNATIVES TO LEAST SQUARES:	
Lnorm_regression	Fits a multiple linear regression model using L_p criteria other than least squares.

CHAPTER 3: CORRELATION AND COVARIANCE

VARIANCES, COVARIANCES, AND CORRELATIONS:	
covariances	Computes the sample variance-covariance or correlation matrix.
partial_covariances	Computes partial covariances or partial correlations from the covariance or correlation matrix.
pooled_covariances	Computes a pooled variance-covariance from the observations.
robust_covariances	Computes a robust estimate of a covariance matrix and mean vector.

CHAPTER 4: ANALYSIS OF VARIANCE AND DESIGNED EXPERIMENTS

GENERAL ANALYSIS OF VARIANCE:	
anova_oneway	Analyzes a one-way classification model.
ancovar	Analyzes a one-way classification model with covariates.
anova_factorial	Analyzes a balanced factorial design with fixed effects.
anova_nested	Analyzes a completely nested random effects model with possibly unequal numbers in the subgroups.
anova_balanced	Analyzes a balanced complete experimental design for a fixed, random, or mixed model.
DESIGNED EXPERIMENTS:	
crd_factorial	Analyzes data from balanced and unbalanced completely randomized experiments.
rcbd_factorial	Analyzes data from balanced and unbalanced randomized complete-block experiments.
latin_square	Analyzes data from latin-square experiments.
lattice	Analyzes balanced and partially-balanced lattice experiments.
split_plot	Analyzes a wide variety of split-plot experiments with fixed, mixed or random factors.
split_split_plot	Analyzes data from split-split-plot experiments.
strip_plot	Analyzes data from strip-plot experiments.
strip_split_plot	Analyzes data from strip-split-plot experiments.
UTILITIES:	
homogeneity	Conducts Bartlett's and Levene's tests of the homogeneity of variance assumption in analysis of variance.
multiple_comparisons	Compares differences among averages using the SNK, LSD, Tukey's, Duncan's and Bonferroni's multiple comparisons tests.
yates	Estimates missing observations in designed experiments using Yate's method.

CHAPTER 5: CATEGORICAL AND DISCRETE DATA ANALYSIS

STATISTICS IN THE TWO-WAY CONTINGENCY TABLE:	
contingency_table	Performs a chi-squared analysis of a two-way contingency table.
exact_enumeration	Computes exact probabilities in a two-way contingency table, using the total enumeration method.
exact_network	Computes exact probabilities in a two-way contingency table using the network algorithm.
GENERALIZED CATEGORICAL MODELS:	
categorical_glm	Analyzes categorical data using logistic, Probit, Poisson, and other generalized linear models.

CHAPTER 6: NONPARAMETRIC STATISTICS

ONE SAMPLE TESTS—NONPARAMETRIC STATISTICS:	
sign_test	Performs a sign test.
wilcoxon_sign_rank	Performs a Wilcoxon signed rank test.
noether_cyclical_trend	Performs the Noether's test for cyclical trend.
cox_stuart_trends_test	Performs the Cox and Stuart sign test for trends in location and dispersion.
tie_statistics	Computes tie statistics for a sample of observations.
TWO OR MORE SAMPLES:	
wilcoxon_rank_sum	Performs a Wilcoxon rank sign test.
kruskal_wallis_test	Performs a Kruskal-Wallis test for identical population medians.
friedmans_test	Performs Friedman's test for a randomized complete block design.
cochran_q_test	Performs Cochran's Q test for related observations.
k_trends_test	Performs k-sample trends test against ordered alternatives.

CHAPTER 7: TESTS OF GOODNESS OF FIT

GENERAL GOODNESS-OF-FIT-TESTS:	
chi_squared_test	Performs a chi-squared goodness-of-fit test.
shapiro_wilk_normality_test	Performs the Shapiro-Wilk test for normality.
lilliefors_normality_test	Performs a Lilliefors test for normality.
chi_squared_normality_test	Performs a chi-squared test for normality.
kolmogorov_one	Performs a Kolmogorov-Smirnov's one-sample test for continuous distributions.
kolmogorov_two	Performs a Kolmogorov-Smirnov's two-sample test.
multivar_normality_test	Computes Mardia's multivariate measures of skewness and kurtosis and tests for multivariate normality.
TESTS FOR RANDOMNESS:	
randomness_test	Performs a test for randomness.

CHAPTER 8: TIME SERIES AND FORECASTING

ARIMA MODELS:	
arma	Computes least-square estimates of parameters for an ARMA model.
max_arma	Exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive, moving average) time series model.
auto_uni_ar	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). Estimates of the autoregressive parameters for the model with minimum AIC are calculated using method of moments, method of least squares or maximum likelihood.
ts_outlier_identification	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.
ts_outlier_forecast	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.
auto_arima	Automatically identifies time series outliers, determines parameters of a multiplicative seasonal ARIMA $(p,0,q)x(0,d,0)_s$ model and produces forecasts that incorporate the effects of outliers whose effects persist beyond the end of the series.

ARIMA MODELS: (CON'T)

arma_forecast	Computes forecasts and their associated probability limits for an ARMA model.
difference	Differences a seasonal or nonseasonal time series.
seasonal_fit	Estimates the optimum seasonality parameters for a time series using an autoregressive model, *AR(p)*, to represent the time series.

MODEL CONSTRUCTION AND EVALUATION UTILITIES:

box_cox_transform	Performs a Box-Cox transformation.
autocorrelation	Computes the sample autocorrelation function of a stationary time series.
crosscorrelation	Computes the sample cross-correlation function of two stationary time series.
multi_crosscorrelation	Computes the multichannel cross-correlation function of two mutually stationary multichannel time series.
partial_autocorrelation	Computes the sample partial autocorrelation function of a stationary time series.
lack_of_fit	Performs lack-of-fit test for an univariate time series or transfer function given the appropriate correlation function.
estimate_missing	Estimates missing values in a time series.

GARCH MODELING:

garch	Computes estimates of the parameters of a Generalized Autoregressive Conditional Heteroskedastic (GARCH)(p, q) model.

FREQUENCY DOMAIN MODELING:

kalman	Performs Kalman filtering and evaluates the likelihood function for the state-space model.

CHAPTER 9: MULTIVARIATE ANALYSIS

HIERARCHICAL CLUSTER ANALYSIS:

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

K-MEANS CLUSTER ANALYSIS:	
cluster_k_means	Performs a *K*-means (centroid) cluster analysis.
PRINCIPAL COMPONENTS:	
principal_components	Computes principal components.
FACTOR ANALYSIS:	
factor_analysis	Extracts initial factor-loading estimates in factor analysis with rotation options.
discriminant_analysis	Performs discriminant function analysis.

CHAPTER 10: SURVIVAL AND RELIABILITY ANALYSIS

SURVIVAL ANALYSIS:	
kaplan_meier_estimates	Computes Kaplan-Meier estimates of survival probabilities in stratified samples.
prop_hazards_gen_lin	Analyzes survival and reliability data using Cox's proportional hazards model.
survival_glm	Analyzes survival data using a generalized linear model.
survival_estimates	Estimates using various parametric models.
RELIABILITY ANALYSIS:	
nonparam_hazard_rate	Estimates a reliability hazard function using a nonparametric approach.
ACTUARIAL TABLES:	
life_tables	Produces population and cohort life tables.

CHAPTER 11: PROBABILITY DISTRIBUTION FUNCTIONS AND INVERSES

DISCRETE RANDOM VARIABLES:	
binomial_cdf	Evaluates the binomial distribution function.
binomial_pdf	Evaluates the binomial probability function.
hypergeometric_cdf	Evaluates the hypergeometric distribution function.

DISCRETE RANDOM VARIABLES: (CON'T)

hypergeometric_pdf	Evaluates the hypergeometric probability function.
poisson_cdf	Evaluates the Poisson distribution function.
poisson_pdf	Evaluates the Poisson probability function.

CONTINUOUS RANDOM VARIABLES:

beta_cdf	Evaluates the beta probability distribution function.
beta_inverse_cdf	Evaluates the inverse of the beta distribution function.
bivariate_normal_cdf	Evaluates the bivariate normal distribution function.
chi_squared_cdf	Evaluates the chi-squared distribution function.
chi_squared_inverse_cdf	Evaluates the inverse of the chi-squared distribution function.
complementary_chi_squared_cdf	Evaluates the complement of the chi-squared distribution.
complementary_F_cdf	Evaluates the complement of the F distribution function.
complementary_t_cdf	Evaluates the complement of the Student's t distribution function.
F_cdf	Evaluates the F distribution function.
F_inverse_cdf	Evaluates the inverse of the F distribution function.
gamma_cdf	Evaluates the gamma distribution function.
gamma_inverse_cdf	Evaluates the inverse of the gamma distribution function.
multivariate_normal_cdf	Evaluates the cumulative distribution function for the multivariate normal distribution.
non_central_chi_sq	Evaluates the noncentral chi-squared distribution function.
non_central_chi_sq_inv	Evaluates the inverse of the noncentral chi-squared function.
non_central_chi_sq_pdf	Evaluates the noncentral chi-squared probability density function.
non_central_F_cdf	Evaluates the noncentral F cumulative distribution function.
non_central_F_inverse_cdf	Evaluates the inverse of the noncentral F cumulative distribution function.
non_central_F_pdf	Evaluates the noncentral F probability density function.

CONTINUOUS RANDOM VARIABLES: (CON'T)

non_central_t_cdf	Evaluates the noncentral Student's t distribution function.
non_central_t_inv_cdf	Evaluates the inverse of the noncentral Student's t distribution function.
non_central_t_pdf	Evaluates the noncentral Student's t probability density function.
normal_cdf	Evaluates the standard normal (Gaussian) distribution function.
normal_inverse_cdf	Evaluates the inverse of the standard normal (Gaussian) distribution function.
t_cdf	Evaluates the Student's t distribution function.
t_inverse_cdf	Evaluates the inverse of the Student's t distribution function.

CHAPTER 12: RANDOM NUMBER GENERATION

UNIVARIATE DISCRETE DISTRIBUTIONS:

random_binomial	Generates pseudorandom binomial numbers from a binomial distribution.
random_geometric	Generates pseudorandom numbers from a geometric distribution.
random_hypergeometric	Generates pseudorandom numbers from a hypergeometric distribution.
random_logarithmic	Generates pseudorandom numbers from a logarithmic distribution.
random_neg_binomial	Generates pseudorandom numbers from a negative binomial distribution.
random_poisson	Generates pseudorandom numbers from a Poisson distribution.
random_uniform_discrete	Generates pseudorandom numbers from a discrete uniform distribution.
random_general_discrete	Generates pseudorandom numbers from a general discrete distribution using an alias method or optionally a table lookup method.

UNIVARIATE CONTINUOUS DISTRIBUTIONS:

random_beta	Generates pseudorandom numbers from a beta distribution.
random_cauchy	Generates pseudorandom numbers from a Cauchy distribution.
random_chi_squared	Generates pseudorandom numbers from a chi-squared distribution.
random_exponential	Generates pseudorandom numbers from a standard exponential distribution.

UNIVARIATE CONTINUOUS DISTRIBUTIONS: (CON'T)

random_exponential_mix	Generates pseudorandom mixed numbers from a standard exponential distribution.
random_gamma	Generates pseudorandom numbers from a standard gamma distribution.
random_lognormal	Generates pseudorandom numbers from a lognormal distribution.
random_normal	Generates pseudorandom numbers from a standard normal distribution.
random_stable	Sets up a table to generate pseudorandom numbers from a general discrete distribution.
random_student_t	Generates pseudorandom numbers from a Student's t distribution.
random_triangular	Generates pseudorandom numbers from a triangular distribution.
random_uniform	Generates pseudorandom numbers from a uniform (0, 1) distribution.
random_von_mises	Generates pseudorandom numbers from a von Mises distribution.
random_weibull	Generates pseudorandom numbers from a Weibull distribution.
random_general_continuous	Generates pseudorandom numbers from a general continuous distribution.
continuous_table_setup	Sets up a table to generate pseudorandom numbers from a general continuous distribution.

MULTIVARIATE CONTINUOUS DISTRIBUTIONS:

random_normal_multivariate	Generates pseudorandom numbers from a multivariate normal distribution.
random_orthogonal_matrix	Generates a pseudorandom orthogonal matrix or a correlation matrix.
random_mvar_from_data	Generates pseudorandom numbers from a multivariate distribution determined from a given sample.
random_multinomial	Generates pseudorandom numbers from a multinomial distribution.
random_sphere	Generates pseudorandom points on a unit circle or K-dimensional sphere.
random_table_twoway	Generates a pseudorandom two-way table.

ORDER STATISTICS:

random_order_normal	Generates pseudorandom order statistics from a standard normal distribution.
random_order_uniform	Generates pseudorandom order statistics from a uniform (0, 1) distribution.

STOCHASTIC PROCESSES:	
random_arma	Generates pseudorandom ARMA process numbers.
random_npp	Generates pseudorandom numbers from a nonhomogeneous Poisson process.
SAMPLES AND PERMUTATIONS:	
random_permutation	Generates a pseudorandom permutation.
random_sample_indices	Generates a simple pseudorandom sample of indices.
random_sample	Generates a simple pseudorandom sample from a finite population.
UTILITY FUNCTIONS:	
random_option	Selects the uniform (0, 1) multiplicative congruential pseudorandom number generator.
random_option_get	Retrieves the uniform (0, 1) multiplicative congruential pseudorandom number generator.
random_seed_get	Retrieves the current value of the seed used in the IMSL random number generators.
random_substream_seed_get	Retrieves a seed for the congruential generators that do not do shuffling that will generate random numbers beginning 100,000 numbers farther along.
random_seed_set	Initializes a random seed for use in the IMSL random number generators.
random_table_set	Sets the current table used in the shuffled generator.
random_table_get	Retrieves the current table used in the shuffled generator.
random_GFSR_table_set	Sets the current table used in the GFSR generator.
random_GFSR_table_get	Retrieves the current table used in the GFSR generator.
random_MT32_init	Initializes the 32-bit Mersenne Twister generator using an array.
random_MT32_table_get	Retrieves the current table used in the 32-bit Mersenne Twister generator.
random_MT32_table_set	Sets the current table used in the 32-bit Mersenne Twister generator.
random_MT64_init	Initializes the 64-bit Mersenne Twister generator using an array.
random_MT64_table_get	Retrieves the current table used in the 64-bit Mersenne Twister generator.
random_MT64_table_set	Sets the current table used in the 64-bit Mersenne Twister generator.

LOW-DISCREPANCY SEQUENCE:	
faure_next_point	Computes a shuffled Faure sequence.

CHAPTER 13: DATA MINING

GENETIC ALGORITHM DATA STRUCTURES:	
ga_chromosome	Creates an *Imsls_f_chromosome* data structure containing unencoded and encoded phenotype information.
ga_copy_chromosome	Copies the contents of one chromosome into another chromosome.
ga_clone_chromosome	Clones an existing chromosome.
ga_individual	Creates an *Imsls_f_individual* data structure from user supplied phenotypes.
ga_copy_individual	Copies the contents of one individual into another individual.
ga_clone_individual	Clones an existing individual.
ga_mutate	Performs the mutation operation on an individual's chromosome.
ga_decode	Decodes an individual's chromosome into its binary, nominal, integer and real phenotypes.
ga_encode	Encodes an individual's binary, nominal, integer and real phenotypes into its chromosome.
ga_free_individual	Frees memory allocated to an existing individual.
ga_population	Creates an *Imsls_f_population* data structure from user supplied individuals.
ga_random_population	Creates an *Imsls_f_population* data structure from randomly generated individuals.
ga_copy_population	Copies the contents of one population into another population.
ga_clone_population	Clones an existing population.
ga_grow_population	Adds individuals to an existing population.
ga_merge_population	Creates a new population by merging two populations with identical chromosome structures.
ga_free_population	Frees memory allocated to an existing population.
genetic_algorithm	Optimizes a user defined fitness function using a tailored genetic algorithm.

NAIVE BAYES:	
naive_bayes_trainer	Trains a Naïve Bayes classifier.
naive_bayes_classification	Classifies unknown patterns using a previously trained Naïve Bayes classifier.
nb_classifier_free	Frees memory allocated to an *Imsls_f_nb_classifier* data structure.
nb_classifier_write	Writes a Naïve Bayes Classifier to an ASCII file for later retrieval using `imsls_f_nb_classifier_read`.
nb_classifier_read	Retrieves a Naïve Bayes Classifier from a file previously saved using `imsls_f_nb_classifier_write`.
NEURAL NETWORK DATA STRUCTURES:	
mlff_network_init	Initializes a data structure for training a neural network.
mlff_network	Multilayered feedforward neural network.
mlff_network_free	Frees memory allocated for an *Imsls_f_NN_Network* data structure.
mlff_network_write	Writes a trained neural network to an ASCII file.
mlff_network_read	Retrieves a neural network from a file previously saved.
mlff_initialize_weights	Initializes weights for multilayered feedforward neural networks prior to network training using one of four user selected methods.
FORECASTING NEURAL NETWORKS:	
mlff_network_trainer	Trains a multilayered feedforward neural network.
mlff_network_forecast	Calculates forecasts for trained multilayered feedforward neural networks.
CLASSIFICATION NEURAL NETWORKS:	
mlff_classification_trainer	Trains a multilayered feedforward neural network for classification.
mlff_pattern_classification	Calculates classifications for trained multilayered feedforward neural networks.
PREPROCESSING FILTERS:	
scale_filter	Scales or unscales continuous data prior to its use in neural network training, testing, or forecasting.
time_series_filter	Converts time series data to the format required for processing by a neural network.

PREPROCESSING FILTERS: (CON'T)

time_series_class_filter	Converts time series data sorted within nominal classes in decreasing chronological order to a useful format for processing by a neural network.
unsupervised_nominal_filter	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.
unsupervised_ordinal_filter	Converts ordinal data into proportions. Optionally, it can also reverse encoding, accepting proportions and converting them into ordinal values.

CHAPTER 14: PRINTING FUNCTIONS

PRINT:	
write_matrix	Prints a rectangular matrix (or vector) stored in contiguous memory locations.
SET:	
page	Sets or retrieves the page width or length.
write_options	Sets or retrieves an option for printing a matrix.

CHAPTER 15: UTILITIES

SET OUPUT FILES:	
output_file	Sets the output file or the error message output file.
version	Returns integer information describing the version of the library, license number, operating system, and compiler.
ERROR HANDLING:	
error_options	Sets various error handling options.
error_type	Gets the type corresponding to the error message from the last function called.
error_message	Gets the text of the error message from the last function called.
error_code	Returns the code corresponding to the error message from the last function called.

C RUNTIME LIBRARY:	
initialize	Initializes the IMSL C Stat Library.
free	Frees memory returned from an IMSL C Stat Library function.
fopen	Opens a file using the C runtime library used by the IMSL C Stat Library.
fclose	Closes a file opened by `imsls_fopen`.
OpenMP:	
omp_options	Sets various OpenMP options.
CONSTANTS:	
machine (integer)	Returns integer information describing the computer's arithmetic.
machine (float)	Returns information describing the computer's floating-point arithmetic.
data_sets	Retrieves a commonly analyzed data set.
MATHEMATICAL SUPPORT:	
mat_mul_rect	Computes the transpose of a matrix, a matrix-vector product, a matrix-matrix product, a bilinear form, or any triple product.
permute_vector	Rearranges the elements of a vector as specified by a permutation.
permute_matrix	Permutes the rows or columns of a matrix.
binomial_coefficient	Evaluates the binomial coefficient.
beta	Evaluates the complete beta function.
beta_incomplete	Evaluates the real incomplete beta function.
log_beta	Evaluates the log of the real beta function.
gamma	Evaluates the real gamma function.
gamma_incomplete	Evaluates the incomplete gamma function.
log_gamma	Evaluates the logarithm of the absolute value of the gamma function.
ctime	Returns the number of CPU seconds used.