

GNU Scientific Library

Reference Manual

Third edition, for GSL Version 1.12

Mark Galassi

Los Alamos National Laboratory

Jim Davies

Department of Computer Science, Georgia Institute of Technology

James Theiler

Astrophysics and Radiation Measurements Group,
Los Alamos National Laboratory

Brian Gough

Network Theory Limited

Gerard Jungman

Theoretical Astrophysics Group, Los Alamos National Laboratory

Patrick Alken

Department of Physics, University of Colorado at Boulder

Michael Booth

Department of Physics and Astronomy, The Johns Hopkins University

Fabrice Rossi

University of Paris-Dauphine

Published by Network Theory Ltd.

A catalogue record for this book is available from the British Library.

Third edition, First printing, January 2009 for version 1.12.

Published by Network Theory Limited.

Email: info@network-theory.co.uk

ISBN: 0-9546120-7-8

This book supersedes the previous edition for version 1.8 (ISBN 0-9541617-3-4).

Original cover design by David Nicholls.

Further information about this book is available from

<http://www.network-theory.co.uk/gsl/manual/>

This book has an unconditional guarantee. If you are not fully satisfied with your purchase for any reason, please contact the publisher at the address above.

Copyright © 2009 Network Theory Ltd. Minor formatting modifications for publication.

Copyright © 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008 The GSL Team.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with the Invariant Sections being “Preface”, “GNU General Public License” and “Free Software Needs Free Documentation”, the Front-Cover text being “A GNU Manual”, and with the Back-Cover Text being (a) (see below). A copy of the license is included in the section entitled “GNU Free Documentation License”.

(a) The Back-Cover Text is: “You have the freedom to copy and modify this GNU Manual.”

The texinfo source files for this manual are available from

<http://www.network-theory.co.uk/gsl/manual/src/>

Table of Contents

Preface	1
1 Introduction.....	3
1.1 Routines available in GSL	3
1.2 GSL is Free Software	4
1.3 Obtaining GSL	4
1.4 No Warranty	5
1.5 Reporting Bugs	5
1.6 Further Information	6
1.7 Conventions used in this manual	6
2 Using the library	7
2.1 An Example Program	7
2.2 Compiling and Linking	7
2.2.1 Linking programs with the library	8
2.2.2 Linking with an alternative BLAS library	8
2.3 Shared Libraries	8
2.4 ANSI C Compliance	9
2.5 Inline functions	9
2.6 Long double	10
2.7 Portability functions	11
2.8 Alternative optimized functions	11
2.9 Support for different numeric types	12
2.10 Compatibility with C++	13
2.11 Aliasing of arrays	13
2.12 Thread-safety	13
2.13 Deprecated Functions	14
2.14 Code Reuse	14
3 Error Handling.....	15
3.1 Error Reporting	15
3.2 Error Codes	16
3.3 Error Handlers	17
3.4 Using GSL error reporting in your own functions	18
3.5 Examples	19

4	Mathematical Functions	21
4.1	Mathematical Constants	21
4.2	Infinites and Not-a-number	22
4.3	Elementary Functions	23
4.4	Small integer powers	24
4.5	Testing the Sign of Numbers	24
4.6	Testing for Odd and Even Numbers	24
4.7	Maximum and Minimum functions	25
4.8	Approximate Comparison of Floating Point Numbers	26
5	Complex Numbers	27
5.1	Representation of complex numbers	27
5.2	Properties of complex numbers	28
5.3	Complex arithmetic operators	28
5.4	Elementary Complex Functions	30
5.5	Complex Trigonometric Functions	30
5.6	Inverse Complex Trigonometric Functions	31
5.7	Complex Hyperbolic Functions	32
5.8	Inverse Complex Hyperbolic Functions	33
5.9	References and Further Reading	34
6	Polynomials	35
6.1	Polynomial Evaluation	35
6.2	Divided Difference Representation of Polynomials	35
6.3	Quadratic Equations	36
6.4	Cubic Equations	37
6.5	General Polynomial Equations	37
6.6	Examples	38
6.7	References and Further Reading	39
7	Special Functions	41
7.1	Usage	41
7.2	The <code>gsl_sf_result</code> struct	42
7.3	Modes	42
7.4	Airy Functions and Derivatives	43
7.4.1	Airy Functions	43
7.4.2	Derivatives of Airy Functions	43
7.4.3	Zeros of Airy Functions	44
7.4.4	Zeros of Derivatives of Airy Functions	44
7.5	Bessel Functions	45
7.5.1	Regular Cylindrical Bessel Functions	45
7.5.2	Irregular Cylindrical Bessel Functions	45
7.5.3	Regular Modified Cylindrical Bessel Functions	46
7.5.4	Irregular Modified Cylindrical Bessel Functions	47
7.5.5	Regular Spherical Bessel Functions	48
7.5.6	Irregular Spherical Bessel Functions	49

7.5.7	Regular Modified Spherical Bessel Functions	49
7.5.8	Irregular Modified Spherical Bessel Functions	50
7.5.9	Regular Bessel Function—Fractional Order	51
7.5.10	Irregular Bessel Functions—Fractional Order	51
7.5.11	Regular Modified Bessel Functions—Fractional Order ..	51
7.5.12	Irregular Modified Bessel Functions—Fractional Order ..	52
7.5.13	Zeros of Regular Bessel Functions	52
7.6	Clausen Functions	53
7.7	Coulomb Functions	53
7.7.1	Normalized Hydrogenic Bound States	53
7.7.2	Coulomb Wave Functions	54
7.7.3	Coulomb Wave Function Normalization Constant	55
7.8	Coupling Coefficients	55
7.8.1	3-j Symbols	55
7.8.2	6-j Symbols	56
7.8.3	9-j Symbols	56
7.9	Dawson Function	56
7.10	Debye Functions	57
7.11	Dilogarithm	58
7.11.1	Real Argument	58
7.11.2	Complex Argument	58
7.12	Elementary Operations	58
7.13	Elliptic Integrals	59
7.13.1	Definition of Legendre Forms	59
7.13.2	Definition of Carlson Forms	59
7.13.3	Legendre Form of Complete Elliptic Integrals	60
7.13.4	Legendre Form of Incomplete Elliptic Integrals	60
7.13.5	Carlson Forms	61
7.14	Elliptic Functions (Jacobi)	62
7.15	Error Functions	62
7.15.1	Error Function	62
7.15.2	Complementary Error Function	62
7.15.3	Log Complementary Error Function	62
7.15.4	Probability functions	62
7.16	Exponential Functions	63
7.16.1	Exponential Function	63
7.16.2	Relative Exponential Functions	64
7.16.3	Exponentiation With Error Estimate	64
7.17	Exponential Integrals	65
7.17.1	Exponential Integral	65
7.17.2	Ei(x)	65
7.17.3	Hyperbolic Integrals	66
7.17.4	Ei ₋₃ (x)	66
7.17.5	Trigonometric Integrals	66
7.17.6	Arctangent Integral	66
7.18	Fermi-Dirac Function	67
7.18.1	Complete Fermi-Dirac Integrals	67
7.18.2	Incomplete Fermi-Dirac Integrals	68

7.19	Gamma and Beta Functions	68
7.19.1	Gamma Functions	68
7.19.2	Factorials	69
7.19.3	Pochhammer Symbol	70
7.19.4	Incomplete Gamma Functions	71
7.19.5	Beta Functions	71
7.19.6	Incomplete Beta Function	72
7.20	Gegenbauer Functions	72
7.21	Hypergeometric Functions	73
7.22	Laguerre Functions	74
7.23	Lambert W Functions	75
7.24	Legendre Functions and Spherical Harmonics	75
7.24.1	Legendre Polynomials	76
7.24.2	Associated Legendre Polynomials and Spherical Harmonics	76
7.24.3	Conical Functions	77
7.24.4	Radial Functions for Hyperbolic Space	78
7.25	Logarithm and Related Functions	79
7.26	Mathieu Functions	80
7.26.1	Mathieu Function Workspace	80
7.26.2	Mathieu Function Characteristic Values	81
7.26.3	Angular Mathieu Functions	81
7.26.4	Radial Mathieu Functions	81
7.27	Power Function	82
7.28	Psi (Digamma) Function	82
7.28.1	Digamma Function	82
7.28.2	Trigamma Function	83
7.28.3	Polygamma Function	83
7.29	Synchrotron Functions	83
7.30	Transport Functions	83
7.31	Trigonometric Functions	84
7.31.1	Circular Trigonometric Functions	84
7.31.2	Trigonometric Functions for Complex Arguments	84
7.31.3	Hyperbolic Trigonometric Functions	85
7.31.4	Conversion Functions	85
7.31.5	Restriction Functions	85
7.31.6	Trigonometric Functions With Error Estimates	85
7.32	Zeta Functions	86
7.32.1	Riemann Zeta Function	86
7.32.2	Riemann Zeta Function Minus One	86
7.32.3	Hurwitz Zeta Function	86
7.32.4	Eta Function	86
7.33	Examples	87
7.34	References and Further Reading	88

8	Vectors and Matrices	89
8.1	Data types	89
8.2	Blocks	89
8.2.1	Block allocation	90
8.2.2	Reading and writing blocks	90
8.2.3	Example programs for blocks	91
8.3	Vectors	91
8.3.1	Vector allocation	92
8.3.2	Accessing vector elements	93
8.3.3	Initializing vector elements	94
8.3.4	Reading and writing vectors	94
8.3.5	Vector views	95
8.3.6	Copying vectors	97
8.3.7	Exchanging elements	98
8.3.8	Vector operations	98
8.3.9	Finding maximum and minimum elements of vectors	98
8.3.10	Vector properties	99
8.3.11	Example programs for vectors	99
8.4	Matrices	101
8.4.1	Matrix allocation	102
8.4.2	Accessing matrix elements	103
8.4.3	Initializing matrix elements	103
8.4.4	Reading and writing matrices	104
8.4.5	Matrix views	104
8.4.6	Creating row and column views	107
8.4.7	Copying matrices	108
8.4.8	Copying rows and columns	109
8.4.9	Exchanging rows and columns	109
8.4.10	Matrix operations	110
8.4.11	Finding maximum and minimum elements of matrices	111
8.4.12	Matrix properties	111
8.4.13	Example programs for matrices	112
8.5	References and Further Reading	115
9	Permutations	117
9.1	The Permutation struct	117
9.2	Permutation allocation	117
9.3	Accessing permutation elements	118
9.4	Permutation properties	118
9.5	Permutation functions	118
9.6	Applying Permutations	119
9.7	Reading and writing permutations	120
9.8	Permutations in cyclic form	121
9.9	Examples	122
9.10	References and Further Reading	124

10	Combinations	125
10.1	The Combination struct	125
10.2	Combination allocation	125
10.3	Accessing combination elements	126
10.4	Combination properties	126
10.5	Combination functions	126
10.6	Reading and writing combinations	127
10.7	Examples	128
10.8	References and Further Reading	129
11	Sorting	131
11.1	Sorting objects	131
11.2	Sorting vectors	132
11.3	Selecting the k smallest or largest elements	133
11.4	Computing the rank	134
11.5	Examples	135
11.6	References and Further Reading	136
12	BLAS Support	137
12.1	GSL BLAS Interface	139
12.1.1	Level 1	139
12.1.2	Level 2	142
12.1.3	Level 3	145
12.2	Examples	150
12.3	References and Further Reading	151
13	Linear Algebra	153
13.1	LU Decomposition	153
13.2	QR Decomposition	155
13.3	QR Decomposition with Column Pivoting	157
13.4	Singular Value Decomposition	159
13.5	Cholesky Decomposition	160
13.6	Tridiagonal Decomposition of Real Symmetric Matrices	161
13.7	Tridiagonal Decomposition of Hermitian Matrices	162
13.8	Hessenberg Decomposition of Real Matrices	163
13.9	Hessenberg-Triangular Decomposition of Real Matrices	164
13.10	Bidiagonalization	164
13.11	Householder Transformations	165
13.12	Householder solver for linear systems	166
13.13	Tridiagonal Systems	166
13.14	Balancing	167
13.15	Examples	168
13.16	References and Further Reading	169

14	Eigensystems	171
14.1	Real Symmetric Matrices	171
14.2	Complex Hermitian Matrices	172
14.3	Real Nonsymmetric Matrices	173
14.4	Real Generalized Symmetric-Definite Eigensystems	175
14.5	Complex Generalized Hermitian-Definite Eigensystems	176
14.6	Real Generalized Nonsymmetric Eigensystems	177
14.7	Sorting Eigenvalues and Eigenvectors	179
14.8	Examples	180
14.9	References and Further Reading	184
15	Fast Fourier Transforms (FFTs)	185
15.1	Mathematical Definitions	185
15.2	Overview of complex data FFTs	186
15.3	Radix-2 FFT routines for complex data	187
15.4	Mixed-radix FFT routines for complex data	190
15.5	Overview of real data FFTs	194
15.6	Radix-2 FFT routines for real data	195
15.7	Mixed-radix FFT routines for real data	197
15.8	References and Further Reading	202
16	Numerical Integration	205
16.1	Introduction	205
16.1.1	Integrands without weight functions	206
16.1.2	Integrands with weight functions	206
16.1.3	Integrands with singular weight functions	206
16.2	QNG non-adaptive Gauss-Kronrod integration	207
16.3	QAG adaptive integration	207
16.4	QAGS adaptive integration with singularities	208
16.5	QAGP adaptive integration with known singular points	209
16.6	QAGI adaptive integration on infinite intervals	209
16.7	QAWC adaptive integration for Cauchy principal values	210
16.8	QAWS adaptive integration for singular functions	211
16.9	QAWO adaptive integration for oscillatory functions	212
16.10	QAWF adaptive integration for Fourier integrals	213
16.11	Error codes	214
16.12	Examples	215
16.13	References and Further Reading	216

17	Random Number Generation	217
17.1	General comments on random numbers	217
17.2	The Random Number Generator Interface	218
17.3	Random number generator initialization	218
17.4	Sampling from a random number generator	219
17.5	Auxiliary random number generator functions	220
17.6	Random number environment variables	221
17.7	Copying random number generator state	222
17.8	Reading and writing random number generator state	222
17.9	Random number generator algorithms	223
17.10	Unix random number generators	227
17.11	Other random number generators	228
17.12	Performance	232
17.13	Examples	232
17.14	References and Further Reading	234
17.15	Acknowledgements	234
18	Quasi-Random Sequences	235
18.1	Quasi-random number generator initialization	235
18.2	Sampling from a quasi-random number generator	235
18.3	Auxiliary quasi-random number generator functions	236
18.4	Saving and resorting quasi-random number generator state ...	236
18.5	Quasi-random number generator algorithms	236
18.6	Examples	237
18.7	References	238
19	Random Number Distributions	239
19.1	Introduction	239
19.2	The Gaussian Distribution	241
19.3	The Gaussian Tail Distribution	243
19.4	The Bivariate Gaussian Distribution	245
19.5	The Exponential Distribution	246
19.6	The Laplace Distribution	247
19.7	The Exponential Power Distribution	248
19.8	The Cauchy Distribution	249
19.9	The Rayleigh Distribution	250
19.10	The Rayleigh Tail Distribution	251
19.11	The Landau Distribution	252
19.12	The Levy alpha-Stable Distributions	253
19.13	The Levy skew alpha-Stable Distribution	254
19.14	The Gamma Distribution	255
19.15	The Flat (Uniform) Distribution	257
19.16	The Lognormal Distribution	258
19.17	The Chi-squared Distribution	259
19.18	The F-distribution	260
19.19	The t-distribution	262

19.20	The Beta Distribution	263
19.21	The Logistic Distribution	264
19.22	The Pareto Distribution	265
19.23	Spherical Vector Distributions.....	266
19.24	The Weibull Distribution	267
19.25	The Type-1 Gumbel Distribution	268
19.26	The Type-2 Gumbel Distribution	269
19.27	The Dirichlet Distribution	270
19.28	General Discrete Distributions	271
19.29	The Poisson Distribution.....	273
19.30	The Bernoulli Distribution	274
19.31	The Binomial Distribution	275
19.32	The Multinomial Distribution	276
19.33	The Negative Binomial Distribution.....	277
19.34	The Pascal Distribution.....	278
19.35	The Geometric Distribution.....	279
19.36	The Hypergeometric Distribution	280
19.37	The Logarithmic Distribution	281
19.38	Shuffling and Sampling	282
19.39	Examples.....	283
19.40	References and Further Reading.....	286
20	Statistics.....	287
20.1	Mean, Standard Deviation and Variance.....	287
20.2	Absolute deviation.....	289
20.3	Higher moments (skewness and kurtosis)	289
20.4	Autocorrelation.....	290
20.5	Covariance.....	291
20.6	Correlation	291
20.7	Weighted Samples	291
20.8	Maximum and Minimum values	294
20.9	Median and Percentiles	295
20.10	Examples.....	296
20.11	References and Further Reading.....	297
21	Histograms	299
21.1	The histogram struct	299
21.2	Histogram allocation.....	300
21.3	Copying Histograms	301
21.4	Updating and accessing histogram elements	301
21.5	Searching histogram ranges	302
21.6	Histogram Statistics	303
21.7	Histogram Operations.....	303
21.8	Reading and writing histograms.....	304
21.9	Resampling from histograms	305
21.10	The histogram probability distribution struct.....	306
21.11	Example programs for histograms.....	307

21.12	Two dimensional histograms	308
21.13	The 2D histogram struct	309
21.14	2D Histogram allocation	310
21.15	Copying 2D Histograms	310
21.16	Updating and accessing 2D histogram elements	311
21.17	Searching 2D histogram ranges	312
21.18	2D Histogram Statistics	312
21.19	2D Histogram Operations	313
21.20	Reading and writing 2D histograms	314
21.21	Resampling from 2D histograms	315
21.22	Example programs for 2D histograms	317
22	N-tuples	319
22.1	The ntuple struct	319
22.2	Creating ntuples	319
22.3	Opening an existing ntuple file	319
22.4	Writing ntuples	320
22.5	Reading ntuples	320
22.6	Closing an ntuple file	320
22.7	Histogramming ntuple values	320
22.8	Examples	321
22.9	References and Further Reading	324
23	Monte Carlo Integration	325
23.1	Interface	325
23.2	PLAIN Monte Carlo	327
23.3	MISER	328
23.4	VEGAS	330
23.5	Examples	333
23.6	References and Further Reading	336
24	Simulated Annealing	337
24.1	Simulated Annealing algorithm	337
24.2	Simulated Annealing functions	338
24.3	Examples	340
24.3.1	Trivial example	340
24.3.2	Traveling Salesman Problem	343
24.4	References and Further Reading	345
25	Ordinary Differential Equations	347
25.1	Defining the ODE System	347
25.2	Stepping Functions	348
25.3	Adaptive Step-size Control	350
25.4	Evolution	352
25.5	Examples	353
25.6	References and Further Reading	357

26	Interpolation	359
26.1	Introduction	359
26.2	Interpolation Functions	359
26.3	Interpolation Types	360
26.4	Index Look-up and Acceleration	361
26.5	Evaluation of Interpolating Functions	361
26.6	Higher-level Interface	362
26.7	Examples	363
26.8	References and Further Reading	366
27	Numerical Differentiation	367
27.1	Functions	367
27.2	Examples	368
27.3	References and Further Reading	369
28	Chebyshev Approximations	371
28.1	Definitions	371
28.2	Creation and Calculation of Chebyshev Series	371
28.3	Auxiliary Functions	372
28.4	Chebyshev Series Evaluation	372
28.5	Derivatives and Integrals	373
28.6	Examples	373
28.7	References and Further Reading	374
29	Series Acceleration	375
29.1	Acceleration functions	375
29.2	Acceleration functions without error estimation	376
29.3	Examples	377
29.4	References and Further Reading	378
30	Wavelet Transforms	379
30.1	Definitions	379
30.2	Initialization	379
30.3	Transform Functions	380
30.3.1	Wavelet transforms in one dimension	381
30.3.2	Wavelet transforms in two dimension	381
30.4	Examples	383
30.5	References and Further Reading	385
31	Discrete Hankel Transforms	387
31.1	Definitions	387
31.2	Functions	388
31.3	References and Further Reading	388

32	One dimensional Root-Finding	389
32.1	Overview	389
32.2	Caveats	390
32.3	Initializing the Solver	390
32.4	Providing the function to solve	391
32.5	Search Bounds and Guesses	394
32.6	Iteration	394
32.7	Search Stopping Parameters	395
32.8	Root Bracketing Algorithms	396
32.9	Root Finding Algorithms using Derivatives	397
32.10	Examples	398
32.11	References and Further Reading	403
33	One dimensional Minimization	405
33.1	Overview	405
33.2	Caveats	406
33.3	Initializing the Minimizer	406
33.4	Providing the function to minimize	407
33.5	Iteration	407
33.6	Stopping Parameters	408
33.7	Minimization Algorithms	409
33.8	Examples	410
33.9	References and Further Reading	411
34	Multidimensional Root-Finding	413
34.1	Overview	413
34.2	Initializing the Solver	414
34.3	Providing the function to solve	415
34.4	Iteration	418
34.5	Search Stopping Parameters	419
34.6	Algorithms using Derivatives	420
34.7	Algorithms without Derivatives	421
34.8	Examples	422
34.9	References and Further Reading	427
35	Multidimensional Minimization	429
35.1	Overview	429
35.2	Caveats	430
35.3	Initializing the Multidimensional Minimizer	430
35.4	Providing a function to minimize	431
35.5	Iteration	433
35.6	Stopping Criteria	434
35.7	Algorithms with Derivatives	435
35.8	Algorithms without Derivatives	436
35.9	Examples	437
35.10	References and Further Reading	441

36	Least-Squares Fitting.....	443
36.1	Overview	443
36.2	Linear regression.....	444
36.3	Linear fitting without a constant term.....	445
36.4	Multi-parameter fitting	446
36.5	Examples.....	448
36.6	References and Further Reading.....	453
37	Nonlinear Least-Squares Fitting.....	455
37.1	Overview	455
37.2	Initializing the Solver	456
37.3	Providing the Function to be Minimized.....	457
37.4	Iteration	458
37.5	Search Stopping Parameters.....	459
37.6	Minimization Algorithms using Derivatives	460
37.7	Minimization Algorithms without Derivatives.....	461
37.8	Computing the covariance matrix of best fit parameters	461
37.9	Examples.....	462
37.10	References and Further Reading.....	467
38	Basis Splines.....	469
38.1	Overview	469
38.2	Initializing the B-splines solver	470
38.3	Constructing the knots vector.....	470
38.4	Evaluation of B-splines.....	471
38.5	Evaluation of B-spline derivatives	471
38.6	Examples.....	472
38.7	References and Further Reading.....	475
39	Physical Constants	477
39.1	Fundamental Constants.....	477
39.2	Astronomy and Astrophysics.....	478
39.3	Atomic and Nuclear Physics.....	478
39.4	Measurement of Time.....	479
39.5	Imperial Units	479
39.6	Speed and Nautical Units.....	480
39.7	Printers Units	480
39.8	Volume, Area and Length	480
39.9	Mass and Weight	481
39.10	Thermal Energy and Power.....	481
39.11	Pressure.....	482
39.12	Viscosity	482
39.13	Light and Illumination.....	482
39.14	Radioactivity	483
39.15	Force and Energy.....	483
39.16	Prefixes.....	483

39.17	Examples	484
39.18	References and Further Reading	485
40	IEEE floating-point arithmetic	487
40.1	Representation of floating point numbers	487
40.2	Setting up your IEEE environment	489
40.3	References and Further Reading	492
Appendix A	Debugging Numerical Programs ..	493
A.1	Using gdb	493
A.2	Examining floating point registers	494
A.3	Handling floating point exceptions	495
A.4	GCC warning options for numerical programs	495
A.5	References and Further Reading	497
Appendix B	Contributors to GSL	499
Appendix C	Autoconf Macros	501
Appendix D	GSL CBLAS Library	503
D.1	Level 1	503
D.2	Level 2	506
D.3	Level 3	512
D.4	Examples	515
Appendix E	GPG verification	517
Free Software Needs Free Documentation		519
Other books from the publisher		521
GNU General Public License		523
GNU Free Documentation License		533
History		539
Function Index		541

Type and Variable Index	561
Concept Index	563

Preface

This manual documents the use of the GNU Scientific Library, a numerical library for C and C++ programmers.

The GNU Scientific Library is *free software*. The term “free software” is sometimes misunderstood—it has nothing to do with price. It is about freedom. It refers to your freedom to run, copy, distribute, study, change and improve the software. With the GNU Scientific Library you have all these freedoms.

The GNU Scientific Library is part of the GNU Project. The GNU Project was launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU system. It was conceived as a way of bringing back the cooperative spirit that prevailed in the computing community in earlier days, by removing the obstacles to cooperation imposed by the owners of proprietary software.

The Free Software Foundation is a tax-exempt charity that raises funds for work on the GNU Project and is dedicated to promoting the freedom to modify and redistribute computer programs. You can support the GNU Project by becoming an associate member of the Free Software Foundation and paying regular membership dues. For more information, visit the website www.fsf.org.

Brian Gough
Publisher
December 2008

1 Introduction

The GNU Scientific Library (GSL) is a collection of routines for numerical computing. The routines have been written from scratch in C, and present a modern Applications Programming Interface (API) for C programmers, allowing wrappers to be written for very high level languages. The source code is distributed under the GNU General Public License.

1.1 Routines available in GSL

The library covers a wide range of topics in numerical computing. Routines are available for the following areas,

Complex Numbers	Roots of Polynomials
Special Functions	Vectors and Matrices
Permutations	Combinations
Sorting	BLAS Support
Linear Algebra	CBLAS Library
Fast Fourier Transforms	Eigensystems
Random Numbers	Quadrature
Random Distributions	Quasi-Random Sequences
Histograms	Statistics
Monte Carlo Integration	N-Tuples
Differential Equations	Simulated Annealing
Numerical Differentiation	Interpolation
Series Acceleration	Chebyshev Approximations
Root-Finding	Discrete Hankel Transforms
Least-Squares Fitting	Minimization
IEEE Floating-Point	Physical Constants
Basis Splines	Wavelets

The use of these routines is described in this manual. Each chapter provides detailed definitions of the functions, followed by example programs and references to the articles on which the algorithms are based.

Where possible the routines have been based on reliable public-domain packages such as FFTPACK and QUADPACK, which the developers of GSL have reimplemented in C with modern coding conventions.

1.2 GSL is Free Software

The subroutines in the GNU Scientific Library are “free software”; this means that everyone is free to use them, and to redistribute them in other free programs. The library is not in the public domain; it is copyrighted and there are conditions on its distribution. These conditions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of the software that they might get from you.

Specifically, we want to make sure that you have the right to share copies of programs that you are given which use the GNU Scientific Library, that you receive their source code or else can get it if you want it, that you can change these programs or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of any code which uses the GNU Scientific Library, you must give the recipients all the rights that you have received. You must make sure that they, too, receive or can get the source code, both to the library and the code which uses it. And you must tell them their rights. This means that the library should not be redistributed in proprietary programs.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the GNU Scientific Library. If these programs are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions for the distribution of software related to the GNU Scientific Library are found in the GNU General Public License (see [GNU General Public License], page 523). Further information about this license is available from the GNU Project webpage *Frequently Asked Questions about the GNU GPL*,

<http://www.gnu.org/copyleft/gpl-faq.html>

The Free Software Foundation also operates a license consulting service for commercial users (contact details available from <http://www.fsf.org/>).

1.3 Obtaining GSL

The source code for the library can be obtained in different ways, by copying it from a friend, purchasing it on CDROM or downloading it from the internet. A list of public ftp servers which carry the source code can be found on the GNU website,

<http://www.gnu.org/software/gsl/>

The preferred platform for the library is a GNU system, which allows it to take advantage of additional features in the GNU C compiler and GNU C library. However, the library is fully portable and should compile on most systems with a C compiler.

Announcements of new releases, updates and other relevant events are made on the `info-gsl@gnu.org` mailing list. To subscribe to this low-volume list, send an email of the following form:

To: `info-gsl-request@gnu.org`
Subject: `subscribe`

You will receive a response asking you to reply in order to confirm your subscription.

1.4 No Warranty

The software described in this manual has no warranty, it is provided “as is”. It is your responsibility to validate the behavior of the routines and their accuracy using the source code provided, or to purchase support and warranties from commercial redistributors. Consult the GNU General Public license for further details (see [GNU General Public License], page 523).

1.5 Reporting Bugs

A list of known bugs can be found in the ‘BUGS’ file included in the GSL distribution or online in the GSL bug tracker.⁽¹⁾ Details of compilation problems can be found in the ‘INSTALL’ file.

If you find a bug which is not listed in these files, please report it to `bug-gsl@gnu.org`.

All bug reports should include:

- The version number of GSL
- The hardware and operating system
- The compiler used, including version number and compilation options
- A description of the bug behavior
- A short program which exercises the bug

It is useful if you can check whether the same problem occurs when the library is compiled without optimization. Thank you.

Any errors or omissions in this manual can also be reported to the same address.

⁽¹⁾ <http://savannah.gnu.org/bugs/?group=gsl>

1.6 Further Information

Additional information, including online copies of this manual, links to related projects, and mailing list archives are available from the website mentioned above.

Any questions about the use and installation of the library can be asked on the mailing list help-gsl@gnu.org. To subscribe to this list, send an email of the following form:

```
To: help-gsl-request@gnu.org
Subject: subscribe
```

This mailing list can be used to ask questions not covered by this manual, and to contact the developers of the library.

If you would like to refer to the GNU Scientific Library in a journal article, the recommended way is to cite this reference manual, e.g. *M. Galassi et al, GNU Scientific Library Reference Manual (3rd Ed.), ISBN 0-9546120-7-8*.

If you want to give a url, use “<http://www.gnu.org/software/gsl/>”.

1.7 Conventions used in this manual

This manual contains many examples which can be typed at the keyboard. A command entered at the terminal is shown like this,

```
$ command
```

The first character on the line is the terminal prompt, and should not be typed. The dollar sign ‘\$’ is used as the standard prompt in this manual, although some systems may use a different character.

The examples assume the use of the GNU operating system. There may be minor differences in the output on other systems. The commands for setting environment variables use the Bourne shell syntax of the standard GNU shell (**bash**).

2 Using the library

This chapter describes how to compile programs that use GSL, and introduces its conventions.

2.1 An Example Program

The following short program demonstrates the use of the library by computing the value of the Bessel function $J_0(x)$ for $x = 5$,

```
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>

int
main (void)
{
    double x = 5.0;
    double y = gsl_sf_bessel_J0 (x);
    printf ("J0(%g) = %.18e\n", x, y);
    return 0;
}
```

The output is shown below, and should be correct to double-precision accuracy,⁽¹⁾

```
J0(5) = -1.775967713143382920e-01
```

The steps needed to compile this program are described in the following sections.

2.2 Compiling and Linking

The library header files are installed in their own ‘gsl’ directory. You should write any preprocessor include statements with a ‘gsl/’ directory prefix thus,

```
#include <gsl/gsl_math.h>
```

If the directory is not installed on the standard search path of your compiler you will also need to provide its location to the preprocessor as a command line flag. The default location of the ‘gsl’ directory is ‘/usr/local/include/gsl’. A typical compilation command for a source file ‘example.c’ with the GNU C compiler gcc is,

```
$ gcc -Wall -I/usr/local/include -c example.c
```

This results in an object file ‘example.o’. The default include path for gcc searches ‘/usr/local/include’ automatically so the -I option can actually be omitted when GSL is installed in its default location.

⁽¹⁾ The last few digits may vary slightly depending on the compiler and platform used—this is normal.

2.2.1 Linking programs with the library

The library is installed as a single file, `'libgsl.a'`. A shared version of the library `'libgsl.so'` is also installed on systems that support shared libraries. The default location of these files is `'/usr/local/lib'`. If this directory is not on the standard search path of your linker you will also need to provide its location as a command line flag.

To link against the library you need to specify both the main library and a supporting CBLAS library, which provides standard basic linear algebra subroutines. A suitable CBLAS implementation is provided in the library `'libgslcblas.a'` if your system does not provide one. The following example shows how to link an application with the library,

```
$ gcc -L/usr/local/lib example.o -lgsl -lgslcblas -lm
```

The default library path for `gcc` searches `'/usr/local/lib'` automatically so the `-L` option can be omitted when GSL is installed in its default location.

2.2.2 Linking with an alternative BLAS library

The following command line shows how you would link the same application with an alternative CBLAS library called `'libcblas'`,

```
$ gcc example.o -lgsl -lcblas -lm
```

For the best performance an optimized platform-specific CBLAS library should be used for `-lcblas`. The library must conform to the CBLAS standard. The ATLAS package provides a portable high-performance BLAS library with a CBLAS interface. It is free software and should be installed for any work requiring fast vector and matrix operations. The following command line will link with the ATLAS library and its CBLAS interface,

```
$ gcc example.o -lgsl -lcblas -latlas -lm
```

If the ATLAS library is installed in a non-standard directory use the `-L` option to add it to the search path, as described above.

For more information about BLAS functions see Chapter 12 [BLAS Support], page 137.

2.3 Shared Libraries

To run a program linked with the shared version of the library the operating system must be able to locate the corresponding `'so'` file at runtime. If the library cannot be found, the following error will occur:

```
$ ./a.out
./a.out: error while loading shared libraries:
libgsl.so.0: cannot open shared object file: No such
file or directory
```

To avoid this error, either modify the system dynamic linker configuration⁽²⁾ or define the shell variable `LD_LIBRARY_PATH` to include the directory where the library is installed.

(2) `'/etc/ld.so.conf'` on GNU/Linux systems.

For example, in the Bourne shell (`/bin/sh` or `/bin/bash`), the library search path can be set with the following commands:

```
$ LD_LIBRARY_PATH=/usr/local/lib
$ export LD_LIBRARY_PATH
$ ./example
```

In the C-shell (`/bin/csh` or `/bin/tcsh`) the equivalent command is,

```
% setenv LD_LIBRARY_PATH /usr/local/lib
```

The standard prompt for the C-shell in the example above is the percent character ‘%’, and should not be typed as part of the command.

To save retyping these commands each session they can be placed in an individual or system-wide login file.

To compile a statically linked version of the program, use the `-static` flag in `gcc`,

```
$ gcc -static example.o -lgsl -lgslcblas -lm
```

2.4 ANSI C Compliance

The library is written in ANSI C and is intended to conform to the ANSI C standard (C89). It should be portable to any system with a working ANSI C compiler.

The library does not rely on any non-ANSI extensions in the interface it exports to the user. Programs you write using GSL can be ANSI compliant. Extensions which can be used in a way compatible with pure ANSI C are supported, however, via conditional compilation. This allows the library to take advantage of compiler extensions on those platforms which support them.

When an ANSI C feature is known to be broken on a particular system the library will exclude any related functions at compile-time. This should make it impossible to link a program that would use these functions and give incorrect results.

To avoid namespace conflicts all exported function names and variables have the prefix `gsl_`, while exported macros have the prefix `GSL_`.

2.5 Inline functions

The `inline` keyword is not part of the original ANSI C standard (C89) so the library does not export any inline function definitions by default. Inline functions were introduced officially in the newer C99 standard but most C89 compilers have also included `inline` as an extension for a long time.

To allow the use of inline functions, the library provides optional inline versions of performance-critical routines by conditional compilation in the exported header files. The inline versions of these functions can be included by defining the macro `HAVE_INLINE` when compiling an application,

```
$ gcc -Wall -c -DHAVE_INLINE example.c
```

If you use `autoconf` this macro can be defined automatically. If you do not define the macro `HAVE_INLINE` then the slower non-inlined versions of the functions will be used instead.

By default, the actual form of the inline keyword is `extern inline`, which is a `gcc` extension that eliminates unnecessary function definitions. If the form `extern inline` causes problems with other compilers a stricter `autoconf` test can be used, see Appendix C [Autoconf Macros], page 501.

When compiling with `gcc` in C99 mode (`gcc -std=c99`) the header files automatically switch to C99-compatible inline function declarations instead of `extern inline`. With other C99 compilers, define the macro `GSL_C99_INLINE` to use these declarations.

2.6 Long double

In general, the algorithms in the library are written for double precision only. The `long double` type is not supported for actual computation.

One reason for this choice is that the precision of `long double` is platform dependent. The IEEE standard only specifies the minimum precision of extended precision numbers, while the precision of `double` is the same on all platforms.

However, it is sometimes necessary to interact with external data in long-double format, so the vector and matrix datatypes include long-double versions.

It should be noted that in some system libraries the `stdio.h` formatted input/output functions `printf` and `scanf` are not implemented correctly for `long double`. Undefined or incorrect results are avoided by testing these functions during the `configure` stage of library compilation and eliminating certain GSL functions which depend on them if necessary. The corresponding line in the `configure` output looks like this,

```
checking whether printf works with long double... no
```

Consequently when `long double` formatted input/output does not work on a given system it should be impossible to link a program which uses GSL functions dependent on this.

If it is necessary to work on a system which does not support formatted `long double` input/output then the options are to use binary formats or to convert `long double` results into `double` for reading and writing.

2.7 Portability functions

To help in writing portable applications GSL provides some implementations of functions that are found in other libraries, such as the BSD math library. You can write your application to use the native versions of these functions, and substitute the GSL versions via a preprocessor macro if they are unavailable on another platform.

For example, after determining whether the BSD function `hypot` is available you can include the following macro definitions in a file ‘`config.h`’ with your application,

```
/* Substitute gsl_hypot for missing system hypot */

#ifdef HAVE_HYPOT
#define hypot gsl_hypot
#endif
```

The application source files can then use the include command `#include <config.h>` to replace each occurrence of `hypot` by `gsl_hypot` when `hypot` is not available. This substitution can be made automatically if you use `autoconf`, see Appendix C [Autoconf Macros], page 501.

In most circumstances the best strategy is to use the native versions of these functions when available, and fall back to GSL versions otherwise, since this allows your application to take advantage of any platform-specific optimizations in the system library. This is the strategy used within GSL itself.

2.8 Alternative optimized functions

The main implementation of some functions in the library will not be optimal on all architectures. For example, there are several ways to compute a Gaussian random variate and their relative speeds are platform-dependent. In cases like this the library provides alternative implementations of these functions with the same interface. If you write your application using calls to the standard implementation you can select an alternative version later via a preprocessor definition. It is also possible to introduce your own optimized functions this way while retaining portability. The following lines demonstrate the use of a platform-dependent choice of methods for sampling from the Gaussian distribution,

```
#ifdef SPARC
#define gsl_ran_gaussian gsl_ran_gaussian_ratio_method
#elif INTEL
#define gsl_ran_gaussian my_gaussian
#endif
```

These lines would be placed in the configuration header file ‘`config.h`’ of the application, which should then be included by all the source files. Note that the alternative implementations will not produce bit-for-bit identical results, and in the case of random number distributions will produce an entirely different stream of random variates.

2.9 Support for different numeric types

Many functions in the library are defined for different numeric types. This feature is implemented by varying the name of the function with a type-related modifier—a primitive form of C++ templates. The modifier is inserted into the function name after the initial module prefix. The following table shows the function names defined for all the numeric types of an imaginary module `gsl_foo` with function `fn`,

<code>gsl_foo_fn</code>	<code>double</code>
<code>gsl_foo_long_double_fn</code>	<code>long double</code>
<code>gsl_foo_float_fn</code>	<code>float</code>
<code>gsl_foo_long_fn</code>	<code>long</code>
<code>gsl_foo_ulong_fn</code>	<code>unsigned long</code>
<code>gsl_foo_int_fn</code>	<code>int</code>
<code>gsl_foo_uint_fn</code>	<code>unsigned int</code>
<code>gsl_foo_short_fn</code>	<code>short</code>
<code>gsl_foo_ushort_fn</code>	<code>unsigned short</code>
<code>gsl_foo_char_fn</code>	<code>char</code>
<code>gsl_foo_uchar_fn</code>	<code>unsigned char</code>

The normal numeric precision `double` is considered the default and does not require a suffix. For example, the function `gsl_stats_mean` computes the mean of double precision numbers, while the function `gsl_stats_int_mean` computes the mean of integers.

A corresponding scheme is used for library defined types, such as `gsl_vector` and `gsl_matrix`. In this case the modifier is appended to the type name. For example, if a module defines a new type-dependent struct or typedef `gsl_foo` it is modified for other types in the following way,

<code>gsl_foo</code>	<code>double</code>
<code>gsl_foo_long_double</code>	<code>long double</code>
<code>gsl_foo_float</code>	<code>float</code>
<code>gsl_foo_long</code>	<code>long</code>
<code>gsl_foo_ulong</code>	<code>unsigned long</code>
<code>gsl_foo_int</code>	<code>int</code>
<code>gsl_foo_uint</code>	<code>unsigned int</code>
<code>gsl_foo_short</code>	<code>short</code>
<code>gsl_foo_ushort</code>	<code>unsigned short</code>
<code>gsl_foo_char</code>	<code>char</code>
<code>gsl_foo_uchar</code>	<code>unsigned char</code>

When a module contains type-dependent definitions the library provides individual header files for each type. The filenames are modified as shown in the below. For convenience the default header includes the definitions for all the types. To include only the double precision header file, or any other specific type, use its individual filename.

<code>#include <gsl/gsl_foo.h></code>	<code>All types</code>
<code>#include <gsl/gsl_foo_double.h></code>	<code>double</code>
<code>#include <gsl/gsl_foo_long_double.h></code>	<code>long double</code>
<code>#include <gsl/gsl_foo_float.h></code>	<code>float</code>
<code>#include <gsl/gsl_foo_long.h></code>	<code>long</code>

```

#include <gsl/gsl_foo_ulong.h>      unsigned long
#include <gsl/gsl_foo_int.h>         int
#include <gsl/gsl_foo_uint.h>        unsigned int
#include <gsl/gsl_foo_short.h>       short
#include <gsl/gsl_foo_ushort.h>      unsigned short
#include <gsl/gsl_foo_char.h>        char
#include <gsl/gsl_foo_uchar.h>       unsigned char

```

2.10 Compatibility with C++

The library header files automatically define functions to have `extern "C"` linkage when included in C++ programs. This allows the functions to be called directly from C++.

To use C++ exception handling within user-defined functions passed to the library as parameters, the library must be built with the additional `CFLAGS` compilation option `'-fexceptions'`.

2.11 Aliasing of arrays

The library assumes that arrays, vectors and matrices passed as modifiable arguments are not aliased and do not overlap with each other. This removes the need for the library to handle overlapping memory regions as a special case, and allows additional optimizations to be used. If overlapping memory regions are passed as modifiable arguments then the results of such functions will be undefined. If the arguments will not be modified (for example, if a function prototype declares them as `const` arguments) then overlapping or aliased memory regions can be safely used.

2.12 Thread-safety

The library can be used in multi-threaded programs. All the functions are thread-safe, in the sense that they do not use static variables. Memory is always associated with objects and not with functions. For functions which use *workspace* objects as temporary storage the workspaces should be allocated on a per-thread basis. For functions which use *table* objects as read-only memory the tables can be used by multiple threads simultaneously. Table arguments are always declared `const` in function prototypes, to indicate that they may be safely accessed by different threads.

There are a small number of static global variables which are used to control the overall behavior of the library (e.g. whether to use range-checking, the function to call on fatal error, etc). These variables are set directly by the user, so they should be initialized once at program startup and not modified by different threads.

2.13 Deprecated Functions

From time to time, it may be necessary for the definitions of some functions to be altered or removed from the library. In these circumstances the functions will first be declared *deprecated* and then removed from subsequent versions of the library. Functions that are deprecated can be disabled in the current release by setting the preprocessor definition `GSL_DISABLE_DEPRECATED`. This allows existing code to be tested for forwards compatibility.

2.14 Code Reuse

Where possible the routines in the library have been written to avoid dependencies between modules and files. This should make it possible to extract individual functions for use in your own applications, without needing to have the whole library installed. You may need to define certain macros such as `GSL_ERROR` and remove some `#include` statements in order to compile the files as standalone units. Reuse of the library code in this way is encouraged, subject to the terms of the GNU General Public License.

3 Error Handling

This chapter describes the way that GSL functions report and handle errors. By examining the status information returned by every function you can determine whether it succeeded or failed, and if it failed you can find out what the precise cause of failure was. You can also define your own error handling functions to modify the default behavior of the library.

The functions described in this section are declared in the header file `'gsl_errno.h'`.

3.1 Error Reporting

The library follows the thread-safe error reporting conventions of the POSIX Threads library. Functions return a non-zero error code to indicate an error and 0 to indicate success.

```
int status = gsl_function (...)

if (status) { /* an error occurred */
    .....
    /* status value specifies the type of error */
}
```

The routines report an error whenever they cannot perform the task requested of them. For example, a root-finding function would return a non-zero error code if could not converge to the requested accuracy, or exceeded a limit on the number of iterations. Situations like this are a normal occurrence when using any mathematical library and you should check the return status of the functions that you call.

Whenever a routine reports an error the return value specifies the type of error. The return value is analogous to the value of the variable `errno` in the C library. The caller can examine the return code and decide what action to take, including ignoring the error if it is not considered serious.

In addition to reporting errors by return codes the library also has an error handler function `gsl_error`. This function is called by other library functions when they report an error, just before they return to the caller. The default behavior of the error handler is to print a message and abort the program,

```
gsl: file.c:67: ERROR: invalid argument supplied by user
Default GSL error handler invoked.
Aborted
```

The purpose of the `gsl_error` handler is to provide a function where a breakpoint can be set that will catch library errors when running under the debugger. It is not intended for use in production programs, which should handle any errors using the return codes.

3.2 Error Codes

The error code numbers returned by library functions are defined in the file ‘`gsl_errno.h`’. They all have the prefix `GSL_` and expand to non-zero constant integer values. Error codes above 1024 are reserved for applications, and are not used by the library. Many of the error codes use the same base name as the corresponding error code in the C library. Here are some of the most common error codes,

`int GSL_EDOM` Macro
 Domain error; used by mathematical functions when an argument value does not fall into the domain over which the function is defined (like `EDOM` in the C library)

`int GSL_ERANGE` Macro
 Range error; used by mathematical functions when the result value is not representable because of overflow or underflow (like `ERANGE` in the C library)

`int GSL_ENOMEM` Macro
 No memory available. The system cannot allocate more virtual memory because its capacity is full (like `ENOMEM` in the C library). This error is reported when a GSL routine encounters problems when trying to allocate memory with `malloc`.

`int GSL_EINVAL` Macro
 Invalid argument. This is used to indicate various kinds of problems with passing the wrong argument to a library function (like `EINVAL` in the C library).

The error codes can be converted into an error message using the function `gsl_strerror`.

`const char * gsl_strerror (const int gsl_errno)` Function
 This function returns a pointer to a string describing the error code `gsl_errno`. For example,

```
printf ("error: %s\n", gsl_strerror (status));
```

would print an error message like **error: output range error** for a status value of `GSL_ERANGE`.

3.3 Error Handlers

The default behavior of the GSL error handler is to print a short message and call `abort`. When this default is in use programs will stop with a core-dump whenever a library routine reports an error. This is intended as a fail-safe default for programs which do not check the return status of library routines (we don't encourage you to write programs this way).

If you turn off the default error handler it is your responsibility to check the return values of routines and handle them yourself. You can also customize the error behavior by providing a new error handler. For example, an alternative error handler could log all errors to a file, ignore certain error conditions (such as underflows), or start the debugger and attach it to the current process when an error occurs.

All GSL error handlers have the type `gsl_error_handler_t`, which is defined in `'gsl_errno.h'`,

`gsl_error_handler_t` Data Type

This is the type of GSL error handler functions. An error handler will be passed four arguments which specify the reason for the error (a string), the name of the source file in which it occurred (also a string), the line number in that file (an integer) and the error number (an integer). The source file and line number are set at compile time using the `__FILE__` and `__LINE__` directives in the preprocessor. An error handler function returns type `void`. Error handler functions should be defined like this,

```
void handler (const char * reason,
              const char * file,
              int line,
              int gsl_errno)
```

To request the use of your own error handler you need to call the function `gsl_set_error_handler` which is also declared in `'gsl_errno.h'`,

`gsl_error_handler_t * gsl_set_error_handler` Function
 (`gsl_error_handler_t * new_handler`)

This function sets a new error handler, *new_handler*, for the GSL library routines. The previous handler is returned (so that you can restore it later). Note that the pointer to a user defined error handler function is stored in a static variable, so there can be only one error handler per program. This function should be not be used in multi-threaded programs except to set up a program-wide error handler from a master thread. The following example shows how to set and restore a new error handler,

```
/* save original handler, install new handler */
old_handler = gsl_set_error_handler (&my_handler);

/* code uses new handler */
.....

/* restore original handler */
gsl_set_error_handler (old_handler);
```

To use the default behavior (abort on error) set the error handler to `NULL`,
`old_handler = gsl_set_error_handler (NULL);`

`gsl_error_handler_t * gsl_set_error_handler_off ()` Function

This function turns off the error handler by defining an error handler which does nothing. This will cause the program to continue after any error, so the return values from any library routines must be checked. This is the recommended behavior for production programs. The previous handler is returned (so that you can restore it later).

The error behavior can be changed for specific applications by recompiling the library with a customized definition of the `GSL_ERROR` macro in the file `'gsl_errno.h'`.

3.4 Using GSL error reporting in your own functions

If you are writing numerical functions in a program which also uses GSL code you may find it convenient to adopt the same error reporting conventions as in the library.

To report an error you need to call the function `gsl_error` with a string describing the error and then return an appropriate error code from `gsl_errno.h`, or a special value, such as `NaN`. For convenience the file `'gsl_errno.h'` defines two macros which carry out these steps:

`GSL_ERROR (reason, gsl_errno)` Macro

This macro reports an error using the GSL conventions and returns a status value of `gsl_errno`. It expands to the following code fragment,

```
gsl_error (reason, __FILE__, __LINE__, gsl_errno);
return gsl_errno;
```

The macro definition in `'gsl_errno.h'` actually wraps the code in a `do { ... } while (0)` block to prevent possible parsing problems.

Here is an example of how the macro could be used to report that a routine did not achieve a requested tolerance. To report the error the routine needs to return the error code `GSL_ETOL`.

```
if (residual > tolerance)
{
    GSL_ERROR("residual exceeds tolerance", GSL_ETOL);
}
```

`GSL_ERROR_VAL (reason, gsl_errno, value)` Macro

This macro is the same as `GSL_ERROR` but returns a user-defined value of `value` instead of an error code. It can be used for mathematical functions that return a floating point value.

The following example shows how to return a NaN at a mathematical singularity using the `GSL_ERROR_VAL` macro,

```
if (x == 0)
{
    GSL_ERROR_VAL("argument lies on singularity",
                  GSL_ERANGE, GSL_NAN);
}
```

3.5 Examples

Here is an example of some code which checks the return value of a function where an error might be reported,

```
#include <stdio.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_fft_complex.h>

...
int status;
size_t n = 37;

gsl_set_error_handler_off();

status = gsl_fft_complex_radix2_forward (data, stride, n);

if (status) {
    if (status == GSL_EINVAL) {
        fprintf (stderr, "invalid argument, n=%d\n", n);
    } else {
        fprintf (stderr, "failed, gsl_errno=%d\n",
                  status);
    }
    exit (-1);
}

...
```

The function `gsl_fft_complex_radix2` only accepts integer lengths which are a power of two. If the variable `n` is not a power of two then the call to the library function will return `GSL_EINVAL`, indicating that the length argument is invalid. The function call to `gsl_set_error_handler_off` stops the default error handler from aborting the program. The `else` clause catches any other possible errors.

4 Mathematical Functions

This chapter describes basic mathematical functions. Some of these functions are present in system libraries, but the alternative versions given here can be used as a substitute when the system functions are not available.

The functions and macros described in this chapter are defined in the header file ‘`gsl_math.h`’.

4.1 Mathematical Constants

The library ensures that the standard BSD mathematical constants are defined. For reference, here is a list of the constants:

M_E

The base of exponentials, e

M_LOG2E

The base-2 logarithm of e , $\log_2(e)$

M_LOG10E

The base-10 logarithm of e , $\log_{10}(e)$

M_SQRT2

The square root of two, $\sqrt{2}$

M_SQRT1_2

The square root of one-half, $\sqrt{1/2}$

M_SQRT3

The square root of three, $\sqrt{3}$

M_PI

The constant pi, π

M_PI_2

Pi divided by two, $\pi/2$

M_PI_4

Pi divided by four, $\pi/4$

M_SQRTPI

The square root of pi, $\sqrt{\pi}$

M_2_SQRTPI

Two divided by the square root of pi, $2/\sqrt{\pi}$

M_1_PI

The reciprocal of pi, $1/\pi$

M_2_PI

Twice the reciprocal of pi, $2/\pi$

M_LN10	The natural logarithm of ten, $\ln(10)$
M_LN2	The natural logarithm of two, $\ln(2)$
M_LNPI	The natural logarithm of pi, $\ln(\pi)$
M_EULER	Euler's constant, γ

4.2 Infinities and Not-a-number

GSL_POSINF	Macro
This macro contains the IEEE representation of positive infinity, $+\infty$. It is computed from the expression <code>+1.0/0.0</code> .	
GSL_NEGINF	Macro
This macro contains the IEEE representation of negative infinity, $-\infty$. It is computed from the expression <code>-1.0/0.0</code> .	
GSL_NAN	Macro
This macro contains the IEEE representation of the Not-a-Number symbol, NaN. It is computed from the ratio <code>0.0/0.0</code> .	
int gsl_isnan (const double x)	Function
This function returns 1 if <i>x</i> is not-a-number.	
int gsl_isinf (const double x)	Function
This function returns +1 if <i>x</i> is positive infinity, -1 if <i>x</i> is negative infinity and 0 otherwise. ⁽¹⁾	
int gsl_finite (const double x)	Function
This function returns 1 if <i>x</i> is a real number, and 0 if it is infinite or not-a-number.	

⁽¹⁾ Note that the C99 standard only requires the system `isinf` function to return a non-zero value, without the sign of the infinity. The implementation in some earlier versions of GSL used the system `isinf` function and may have this behavior on some platforms. Therefore, it is advisable to test the sign of *x* separately, if needed, rather than relying the sign of the return value from `gsl_isinf()`.

4.3 Elementary Functions

The following routines provide portable implementations of functions found in the BSD math library. When native versions are not available the functions described here can be used instead. The substitution can be made automatically if you use `autoconf` to compile your application (see Section 2.7 [Portability functions], page 11).

`double gsl_log1p (const double x)` Function
 This function computes the value of $\log(1 + x)$ in a way that is accurate for small x . It provides an alternative to the BSD math function `log1p(x)`.

`double gsl_expml (const double x)` Function
 This function computes the value of $\exp(x) - 1$ in a way that is accurate for small x . It provides an alternative to the BSD math function `expml(x)`.

`double gsl_hypot (const double x, const double y)` Function
 This function computes the value of $\sqrt{x^2 + y^2}$ in a way that avoids overflow. It provides an alternative to the BSD math function `hypot(x, y)`.

`double gsl_hypot3 (const double x, const double y, const double z)` Function
 This function computes the value of $\sqrt{x^2 + y^2 + z^2}$ in a way that avoids overflow.

`double gsl_acosh (const double x)` Function
 This function computes the value of $\operatorname{arccosh}(x)$. It provides an alternative to the standard math function `acosh(x)`.

`double gsl_asinh (const double x)` Function
 This function computes the value of $\operatorname{arcsinh}(x)$. It provides an alternative to the standard math function `asinh(x)`.

`double gsl_atanh (const double x)` Function
 This function computes the value of $\operatorname{arctanh}(x)$. It provides an alternative to the standard math function `atanh(x)`.

`double gsl_ldexp (double x, int e)` Function
 This function computes the value of $x * 2^e$. It provides an alternative to the standard math function `ldexp(x, e)`.

`double gsl_frexp (double x, int * e)` Function
 This function splits the number x into its normalized fraction f and exponent e , such that $x = f * 2^e$ and $0.5 \leq f < 1$. The function returns f and stores the exponent in e . If x is zero, both f and e are set to zero. This function provides an alternative to the standard math function `frexp(x, e)`.

4.4 Small integer powers

A common complaint about the standard C library is its lack of a function for calculating (small) integer powers. GSL provides some simple functions to fill this gap. For reasons of efficiency, these functions do not check for overflow or underflow conditions.

double gsl_pow_int (double x, int n) Function
 This routine computes the power x^n for integer n . The power is computed efficiently—for example, x^8 is computed as $((x^2)^2)^2$, requiring only 3 multiplications. A version of this function which also computes the numerical error in the result is available as **gsl_sf_pow_int_e**.

double gsl_pow_2 (const double x) Function
double gsl_pow_3 (const double x) Function
double gsl_pow_4 (const double x) Function
double gsl_pow_5 (const double x) Function
double gsl_pow_6 (const double x) Function
double gsl_pow_7 (const double x) Function
double gsl_pow_8 (const double x) Function
double gsl_pow_9 (const double x) Function

These functions can be used to compute small integer powers x^2 , x^3 , etc. efficiently. The functions will be inlined when **HAVE_INLINE** is defined, so that use of these functions should be as efficient as explicitly writing the corresponding product expression.

```
#include <gsl/gsl_math.h>
double y = gsl_pow_4 (3.141) /* compute 3.141**4 */
```

4.5 Testing the Sign of Numbers

GSL_SIGN (x) Macro
 This macro returns the sign of x . It is defined as $((x) >= 0 ? 1 : -1)$. Note that with this definition the sign of zero is positive (regardless of its IEEE sign bit).

4.6 Testing for Odd and Even Numbers

GSL_IS_ODD (n) Macro
 This macro evaluates to 1 if n is odd and 0 if n is even. The argument n must be of integer type.

GSL_IS_EVEN (n) Macro
 This macro is the opposite of **GSL_IS_ODD(n)**. It evaluates to 1 if n is even and 0 if n is odd. The argument n must be of integer type.

4.7 Maximum and Minimum functions

Note that the following macros perform multiple evaluations of their arguments, so they should not be used with arguments that have side effects (such as a call to a random number generator).

GSL_MAX (a, b) Macro
 This macro returns the maximum of *a* and *b*. It is defined as ((*a*) > (*b*) ? (*a*):(*b*)).

GSL_MIN (a, b) Macro
 This macro returns the minimum of *a* and *b*. It is defined as ((*a*) < (*b*) ? (*a*):(*b*)).

extern inline double GSL_MAX_DBL (double a, double b) Function
 This function returns the maximum of the double precision numbers *a* and *b* using an inline function. The use of a function allows for type checking of the arguments as an extra safety feature. On platforms where inline functions are not available the macro **GSL_MAX** will be automatically substituted.

extern inline double GSL_MIN_DBL (double a, double b) Function
 This function returns the minimum of the double precision numbers *a* and *b* using an inline function. The use of a function allows for type checking of the arguments as an extra safety feature. On platforms where inline functions are not available the macro **GSL_MIN** will be automatically substituted.

extern inline int GSL_MAX_INT (int a, int b) Function
extern inline int GSL_MIN_INT (int a, int b) Function
 These functions return the maximum or minimum of the integers *a* and *b* using an inline function. On platforms where inline functions are not available the macros **GSL_MAX** or **GSL_MIN** will be automatically substituted.

extern inline long double GSL_MAX_LDBL (long double a, long double b) Function

extern inline long double GSL_MIN_LDBL (long double a, long double b) Function

These functions return the maximum or minimum of the long doubles *a* and *b* using an inline function. On platforms where inline functions are not available the macros **GSL_MAX** or **GSL_MIN** will be automatically substituted.

4.8 Approximate Comparison of Floating Point Numbers

It is sometimes useful to be able to compare two floating point numbers approximately, to allow for rounding and truncation errors. The following function implements the approximate floating-point comparison algorithm proposed by D.E. Knuth in Section 4.2.2 of *Seminumerical Algorithms* (3rd edition).

`int gsl_fcmp (double x, double y, double epsilon)` Function

This function determines whether x and y are approximately equal to a relative accuracy ϵ .

The relative accuracy is measured using an interval of size 2δ , where $\delta = 2^k \epsilon$ and k is the maximum base-2 exponent of x and y as computed by the function `frexp`.

If x and y lie within this interval, they are considered approximately equal and the function returns 0. Otherwise if $x < y$, the function returns -1 , or if $x > y$, the function returns $+1$.

Note that x and y are compared to relative accuracy, so this function is not suitable for testing whether a value is approximately zero.

The implementation is based on the package `fcmp` by T.C. Belding.

5 Complex Numbers

The functions described in this chapter provide support for complex numbers. The algorithms take care to avoid unnecessary intermediate underflows and overflows, allowing the functions to be evaluated over as much of the complex plane as possible.

For multiple-valued functions the branch cuts have been chosen to follow the conventions of Abramowitz and Stegun in the *Handbook of Mathematical Functions*. The functions return principal values which are the same as those in GNU Calc, which in turn are the same as those in *Common Lisp, The Language (Second Edition)*⁽¹⁾ and the HP-28/48 series of calculators.

The complex types are defined in the header file ‘`gsl_complex.h`’, while the corresponding complex functions and arithmetic operations are defined in ‘`gsl_complex_math.h`’.

5.1 Representation of complex numbers

Complex numbers are represented using the type `gsl_complex`. The internal representation of this type may vary across platforms and should not be accessed directly. The functions and macros described below allow complex numbers to be manipulated in a portable way.

For reference, the default form of the `gsl_complex` type is given by the following struct,

```
typedef struct
{
    double dat[2];
} gsl_complex;
```

The real and imaginary part are stored in contiguous elements of a two element array. This eliminates any padding between the real and imaginary parts, `dat[0]` and `dat[1]`, allowing the struct to be mapped correctly onto packed complex arrays.

`gsl_complex gsl_complex_rect (double x, double y)` Function

This function uses the rectangular cartesian components (x, y) to return the complex number $z = x + iy$. An inline version of this function is used when `HAVE_INLINE` is defined.

`gsl_complex gsl_complex_polar (double r, double theta)` Function

This function returns the complex number $z = r \exp(i\theta) = r(\cos(\theta) + i \sin(\theta))$ from the polar representation (r, θ) .

`GSL_REAL (z)` Macro

`GSL_IMAG (z)` Macro

These macros return the real and imaginary parts of the complex number z .

⁽¹⁾ Note that the first edition uses different definitions.

GSL_SET_COMPLEX (*zp*, *x*, *y*) Macro

This macro uses the cartesian components (*x*,*y*) to set the real and imaginary parts of the complex number pointed to by *zp*. For example,

```
GSL_SET_COMPLEX(&z, 3, 4)
```

sets *z* to be $3 + 4i$.

GSL_SET_REAL (*zp*,*x*) Macro

GSL_SET_IMAG (*zp*,*y*) Macro

These macros allow the real and imaginary parts of the complex number pointed to by *zp* to be set independently.

5.2 Properties of complex numbers

double gsl_complex_arg (**gsl_complex** *z*) Function

This function returns the argument of the complex number *z*, $\arg(z)$, where $-\pi < \arg(z) \leq \pi$.

double gsl_complex_abs (**gsl_complex** *z*) Function

This function returns the magnitude of the complex number *z*, $|z|$.

double gsl_complex_abs2 (**gsl_complex** *z*) Function

This function returns the squared magnitude of the complex number *z*, $|z|^2$.

double gsl_complex_logabs (**gsl_complex** *z*) Function

This function returns the natural logarithm of the magnitude of the complex number *z*, $\log |z|$. It allows an accurate evaluation of $\log |z|$ when $|z|$ is close to one. The direct evaluation of $\log(\text{gsl_complex_abs}(z))$ would lead to a loss of precision in this case.

5.3 Complex arithmetic operators

gsl_complex gsl_complex_add (**gsl_complex** *a*, **gsl_complex** *b*) Function

This function returns the sum of the complex numbers *a* and *b*, $z = a + b$.

gsl_complex gsl_complex_sub (**gsl_complex** *a*, **gsl_complex** *b*) Function

This function returns the difference of the complex numbers *a* and *b*, $z = a - b$.

gsl_complex gsl_complex_mul (**gsl_complex** *a*, **gsl_complex** *b*) Function

This function returns the product of the complex numbers *a* and *b*, $z = ab$.

gsl_complex gsl_complex_div (**gsl_complex** *a*, **gsl_complex** *b*) Function

This function returns the quotient of the complex numbers *a* and *b*, $z = a/b$.

gsl_complex gsl_complex_add_real (**gsl_complex** *a*, **double** *x*) Function

This function returns the sum of the complex number *a* and the real number *x*, $z = a + x$.

Function Index

C

cblas_caxpy	504	cblas_drotm	505
cblas_ccopy	504	cblas_drotmg	505
cblas_cdotc_sub	503	cblas_dsbmv	510
cblas_cdotu_sub	503	cblas_dscal	505
cblas_cgbmv	507	cblas_dsdot	503
cblas_cgemm	513	cblas_dspmv	510
cblas_cgemv	507	cblas_dspr	510
cblas_cgerc	511	cblas_dspr2	510
cblas_cgeru	511	cblas_dswap	504
cblas_chbmv	510	cblas_dsymm	513
cblas_chemm	515	cblas_dsymv	510
cblas_chemv	510	cblas_dsyr	510
cblas_cher	511	cblas_dsyr2	510
cblas_cher2	511	cblas_dsyr2k	513
cblas_cher2k	515	cblas_dsyrok	513
cblas_cherk	515	cblas_dtbmv	507
cblas_chpmv	510	cblas_dtbsv	507
cblas_chpr	511	cblas_dtpmv	507
cblas_chpr2	511	cblas_dtpsv	507
cblas_cscal	505	cblas_dtrmm	513
cblas_cswap	504	cblas_dtrmv	507
cblas_csymm	513	cblas_dtrsm	513
cblas_csyr2k	514	cblas_dtrsv	507
cblas_csyrok	513	cblas_dzasum	504
cblas_ctbmv	508	cblas_dznrm2	503
cblas_ctbsv	508	cblas_icamax	504
cblas_ctpmv	508	cblas_idamax	504
cblas_ctpsv	508	cblas_isamax	504
cblas_ctrmm	514	cblas_izamax	504
cblas_ctrsm	514	cblas_sasum	503
cblas_ctrsv	508	cblas_saxpy	504
cblas_dasum	503	cblas_scasum	503
cblas_daxpy	504	cblas_scnrm2	503
cblas_dcopy	504	cblas_scopy	504
cblas_ddot	503	cblas_sdot	503
cblas_dgbmv	507	cblas_sdsdot	503
cblas_dgemm	513	cblas_sgbmv	506
cblas_dgemv	506	cblas_sgemm	512
cblas_dger	510	cblas_sgemv	506
cblas_dnrm2	503	cblas_sger	509
cblas_drot	505	cblas_snrm2	503
cblas_drotg	505	cblas_srot	505
		cblas_srotg	504
		cblas_srotm	505
		cblas_srotmg	504

<code>cblas_ssbmv</code>	509	<code>cblas_ztpsv</code>	509
<code>cblas_sscal</code>	505	<code>cblas_ztrmm</code>	514
<code>cblas_sspmv</code>	509	<code>cblas_ztrmv</code>	508
<code>cblas_sspr</code>	509	<code>cblas_ztrsm</code>	514
<code>cblas_sspr2</code>	509	<code>cblas_ztrsx</code>	509
<code>cblas_sswap</code>	504		
<code>cblas_ssymm</code>	512		
<code>cblas_ssymv</code>	509		
<code>cblas_ssyr</code>	509		
<code>cblas_ssyr2</code>	509		
<code>cblas_ssyr2k</code>	512		
<code>cblas_ssyrk</code>	512		
<code>cblas_stbmv</code>	506		
<code>cblas_stbsv</code>	506		
<code>cblas_stpmv</code>	506		
<code>cblas_stpsv</code>	506		
<code>cblas_strmm</code>	512		
<code>cblas_strmv</code>	506		
<code>cblas_strsm</code>	512		
<code>cblas_strsv</code>	506		
<code>cblas_xerbla</code>	515		
<code>cblas_zaxpy</code>	504		
<code>cblas_zcopy</code>	504		
<code>cblas_zdotc_sub</code>	503		
<code>cblas_zdotu_sub</code>	503		
<code>cblas_zdscal</code>	505		
<code>cblas_zgbmv</code>	508		
<code>cblas_zgemm</code>	514		
<code>cblas_zgemv</code>	508		
<code>cblas_zgerc</code>	511		
<code>cblas_zgeru</code>	511		
<code>cblas_zhbmj</code>	511		
<code>cblas_zhemm</code>	515		
<code>cblas_zhemv</code>	511		
<code>cblas_zher</code>	511		
<code>cblas_zher2</code>	512		
<code>cblas_zher2k</code>	515		
<code>cblas_zherk</code>	515		
<code>cblas_zhpmv</code>	511		
<code>cblas_zhpr</code>	512		
<code>cblas_zhpr2</code>	512		
<code>cblas_zscal</code>	505		
<code>cblas_zswap</code>	504		
<code>cblas_zsymm</code>	514		
<code>cblas_zsyr2k</code>	514		
<code>cblas_zsyrk</code>	514		
<code>cblas_ztbmv</code>	508		
<code>cblas_ztbsv</code>	509		
<code>cblas_ztpmv</code>	508		

G

<code>gsl_acosh</code>	23
<code>gsl_asinh</code>	23
<code>gsl_atanh</code>	23
<code>gsl_blas_caxpy</code>	140
<code>gsl_blas_ccopy</code>	140
<code>gsl_blas_cdotc</code>	139
<code>gsl_blas_cdotu</code>	139
<code>gsl_blas_cgemm</code>	145
<code>gsl_blas_cgemv</code>	142
<code>gsl_blas_cgerc</code>	144
<code>gsl_blas_cgeru</code>	144
<code>gsl_blas_chemm</code>	146
<code>gsl_blas_chemv</code>	143
<code>gsl_blas_cher</code>	144
<code>gsl_blas_cher2</code>	145
<code>gsl_blas_cher2k</code>	149
<code>gsl_blas_cherk</code>	148
<code>gsl_blas_cscal</code>	141
<code>gsl_blas_csscal</code>	141
<code>gsl_blas_cswap</code>	140
<code>gsl_blas_csymm</code>	146
<code>gsl_blas_csyr2k</code>	149
<code>gsl_blas_csyrrk</code>	148
<code>gsl_blas_ctrmm</code>	147
<code>gsl_blas_ctrsm</code>	142
<code>gsl_blas_ctrsv</code>	143
<code>gsl_blas_daxpy</code>	139
<code>gsl_blas_daxpy</code>	140
<code>gsl_blas_dcopy</code>	140
<code>gsl_blas_ddot</code>	139
<code>gsl_blas_dgemm</code>	145
<code>gsl_blas_dgemv</code>	142
<code>gsl_blas_dger</code>	144
<code>gsl_blas_dnrm2</code>	139
<code>gsl_blas_drot</code>	141
<code>gsl_blas_drotg</code>	141
<code>gsl_blas_drotm</code>	141
<code>gsl_blas_drotmg</code>	141
<code>gsl_blas_dscal</code>	141
<code>gsl_blas_dsdot</code>	139

<code>gsl_blas_dswap</code>	140	<code>gsl_blas_zgemm</code>	145
<code>gsl_blas_dsymm</code>	146	<code>gsl_blas_zgemv</code>	142
<code>gsl_blas_dsymv</code>	143	<code>gsl_blas_zgerc</code>	144
<code>gsl_blas_dsyr</code>	144	<code>gsl_blas_zgeru</code>	144
<code>gsl_blas_dsyr2</code>	145	<code>gsl_blas_zhemm</code>	146
<code>gsl_blas_dsyr2k</code>	149	<code>gsl_blas_zhemv</code>	143
<code>gsl_blas_dsyrrk</code>	148	<code>gsl_blas_zher</code>	144
<code>gsl_blas_dtrmm</code>	147	<code>gsl_blas_zher2</code>	145
<code>gsl_blas_dtrmv</code>	142	<code>gsl_blas_zher2k</code>	149
<code>gsl_blas_dtrsm</code>	147	<code>gsl_blas_zherk</code>	148
<code>gsl_blas_dtrsv</code>	143	<code>gsl_blas_zscal</code>	141
<code>gsl_blas_dzasum</code>	140	<code>gsl_blas_zswap</code>	140
<code>gsl_blas_dznrm2</code>	139	<code>gsl_blas_zsymm</code>	146
<code>gsl_blas_icamax</code>	140	<code>gsl_blas_zsyr2k</code>	149
<code>gsl_blas_idamax</code>	140	<code>gsl_blas_zsyrk</code>	148
<code>gsl_blas_isamax</code>	140	<code>gsl_blas_ztrmm</code>	147
<code>gsl_blas_izamax</code>	140	<code>gsl_blas_ztrmv</code>	142
<code>gsl_blas_sasum</code>	139	<code>gsl_blas_ztrsm</code>	147
<code>gsl_blas_saxpy</code>	140	<code>gsl_blas_ztrsv</code>	143
<code>gsl_blas_scasum</code>	140	<code>gsl_block_alloc</code>	90
<code>gsl_blas_scnrm2</code>	139	<code>gsl_block_calloc</code>	90
<code>gsl_blas_scopy</code>	140	<code>gsl_block_fprintf</code>	91
<code>gsl_blas_sdot</code>	139	<code>gsl_block_fread</code>	90
<code>gsl_blas_sdsdot</code>	139	<code>gsl_block_free</code>	90
<code>gsl_blas_sgemm</code>	145	<code>gsl_block_fscanf</code>	91
<code>gsl_blas_sgemv</code>	142	<code>gsl_block_fwrite</code>	90
<code>gsl_blas_sger</code>	144	<code>gsl_bspline_alloc</code>	470
<code>gsl_blas_snrm2</code>	139	<code>gsl_bspline_deriv_alloc</code>	470
<code>gsl_blas_srot</code>	141	<code>gsl_bspline_deriv_eval</code>	471
<code>gsl_blas_srotg</code>	141	<code>gsl_bspline_deriv_eval_nonzero</code> ..	471
<code>gsl_blas_srotm</code>	141	<code>gsl_bspline_deriv_free</code>	470
<code>gsl_blas_srotmg</code>	141	<code>gsl_bspline_eval</code>	471
<code>gsl_blas_sscal</code>	141	<code>gsl_bspline_eval_nonzero</code>	471
<code>gsl_blas_sswap</code>	140	<code>gsl_bspline_free</code>	470
<code>gsl_blas_ssymm</code>	146	<code>gsl_bspline_knots</code>	470
<code>gsl_blas_ssymv</code>	143	<code>gsl_bspline_knots_uniform</code>	470
<code>gsl_blas_ssyr</code>	144	<code>gsl_bspline_ncoeffs</code>	471
<code>gsl_blas_ssyr2</code>	145	<code>gsl_cdf_beta_P</code>	263
<code>gsl_blas_ssyr2k</code>	149	<code>gsl_cdf_beta_Pinv</code>	263
<code>gsl_blas_ssyrk</code>	148	<code>gsl_cdf_beta_Q</code>	263
<code>gsl_blas_strmm</code>	147	<code>gsl_cdf_beta_Qinv</code>	263
<code>gsl_blas_strmv</code>	142	<code>gsl_cdf_binomial_P</code>	275
<code>gsl_blas_strsm</code>	147	<code>gsl_cdf_binomial_Q</code>	275
<code>gsl_blas_strsv</code>	143	<code>gsl_cdf_cauchy_P</code>	249
<code>gsl_blas_zaxpy</code>	140	<code>gsl_cdf_cauchy_Pinv</code>	249
<code>gsl_blas_zcopy</code>	140	<code>gsl_cdf_cauchy_Q</code>	249
<code>gsl_blas_zdotc</code>	139	<code>gsl_cdf_cauchy_Qinv</code>	249
<code>gsl_blas_zdotu</code>	139	<code>gsl_cdf_chisq_P</code>	259
<code>gsl_blas_zdscal</code>	141	<code>gsl_cdf_chisq_Pinv</code>	259

<code>gsl_cdf_chisq_Q</code>	259	<code>gsl_cdf_negative_binomial_Q</code>	277
<code>gsl_cdf_chisq_Qinv</code>	259	<code>gsl_cdf_pareto_P</code>	265
<code>gsl_cdf_exponential_P</code>	246	<code>gsl_cdf_pareto_Pinv</code>	265
<code>gsl_cdf_exponential_Pinv</code>	246	<code>gsl_cdf_pareto_Q</code>	265
<code>gsl_cdf_exponential_Q</code>	246	<code>gsl_cdf_pareto_Qinv</code>	265
<code>gsl_cdf_exponential_Qinv</code>	246	<code>gsl_cdf_pascal_P</code>	278
<code>gsl_cdf_exppow_P</code>	248	<code>gsl_cdf_pascal_Q</code>	278
<code>gsl_cdf_exppow_Q</code>	248	<code>gsl_cdf_poisson_P</code>	273
<code>gsl_cdf_fdist_P</code>	261	<code>gsl_cdf_poisson_Q</code>	273
<code>gsl_cdf_fdist_Pinv</code>	261	<code>gsl_cdf_rayleigh_P</code>	250
<code>gsl_cdf_fdist_Q</code>	261	<code>gsl_cdf_rayleigh_Pinv</code>	250
<code>gsl_cdf_fdist_Qinv</code>	261	<code>gsl_cdf_rayleigh_Q</code>	250
<code>gsl_cdf_flat_P</code>	257	<code>gsl_cdf_rayleigh_Qinv</code>	250
<code>gsl_cdf_flat_Pinv</code>	257	<code>gsl_cdf_tdist_P</code>	262
<code>gsl_cdf_flat_Q</code>	257	<code>gsl_cdf_tdist_Pinv</code>	262
<code>gsl_cdf_flat_Qinv</code>	257	<code>gsl_cdf_tdist_Q</code>	262
<code>gsl_cdf_gamma_P</code>	256	<code>gsl_cdf_tdist_Qinv</code>	262
<code>gsl_cdf_gamma_Pinv</code>	256	<code>gsl_cdf_ugaussian_P</code>	242
<code>gsl_cdf_gamma_Q</code>	256	<code>gsl_cdf_ugaussian_Pinv</code>	242
<code>gsl_cdf_gamma_Qinv</code>	256	<code>gsl_cdf_ugaussian_Q</code>	242
<code>gsl_cdf_gaussian_P</code>	242	<code>gsl_cdf_ugaussian_Qinv</code>	242
<code>gsl_cdf_gaussian_Pinv</code>	242	<code>gsl_cdf_weibull_P</code>	267
<code>gsl_cdf_gaussian_Q</code>	242	<code>gsl_cdf_weibull_Pinv</code>	267
<code>gsl_cdf_gaussian_Qinv</code>	242	<code>gsl_cdf_weibull_Q</code>	267
<code>gsl_cdf_geometric_P</code>	279	<code>gsl_cdf_weibull_Qinv</code>	267
<code>gsl_cdf_geometric_Q</code>	279	<code>gsl_cheb_alloc</code>	371
<code>gsl_cdf_gumbel1_P</code>	268	<code>gsl_cheb_calc_deriv</code>	373
<code>gsl_cdf_gumbel1_Pinv</code>	268	<code>gsl_cheb_calc_integ</code>	373
<code>gsl_cdf_gumbel1_Q</code>	268	<code>gsl_cheb_coeffs</code>	372
<code>gsl_cdf_gumbel1_Qinv</code>	268	<code>gsl_cheb_eval</code>	372
<code>gsl_cdf_gumbel2_P</code>	269	<code>gsl_cheb_eval_err</code>	372
<code>gsl_cdf_gumbel2_Pinv</code>	269	<code>gsl_cheb_eval_n</code>	372
<code>gsl_cdf_gumbel2_Q</code>	269	<code>gsl_cheb_eval_n_err</code>	372
<code>gsl_cdf_gumbel2_Qinv</code>	269	<code>gsl_cheb_free</code>	371
<code>gsl_cdf_hypergeometric_P</code>	280	<code>gsl_cheb_init</code>	371
<code>gsl_cdf_hypergeometric_Q</code>	280	<code>gsl_cheb_order</code>	372
<code>gsl_cdf_laplace_P</code>	247	<code>gsl_cheb_size</code>	372
<code>gsl_cdf_laplace_Pinv</code>	247	<code>gsl_combination_alloc</code>	125
<code>gsl_cdf_laplace_Q</code>	247	<code>gsl_combination_calloc</code>	125
<code>gsl_cdf_laplace_Qinv</code>	247	<code>gsl_combination_data</code>	126
<code>gsl_cdf_logistic_P</code>	264	<code>gsl_combination_fprintf</code>	127
<code>gsl_cdf_logistic_Pinv</code>	264	<code>gsl_combination_fread</code>	127
<code>gsl_cdf_logistic_Q</code>	264	<code>gsl_combination_free</code>	125
<code>gsl_cdf_logistic_Qinv</code>	264	<code>gsl_combination_fscanf</code>	127
<code>gsl_cdf_lognormal_P</code>	258	<code>gsl_combination_fwrite</code>	127
<code>gsl_cdf_lognormal_Pinv</code>	258	<code>gsl_combination_get</code>	126
<code>gsl_cdf_lognormal_Q</code>	258	<code>gsl_combination_init_first</code>	125
<code>gsl_cdf_lognormal_Qinv</code>	258	<code>gsl_combination_init_last</code>	125
<code>gsl_cdf_negative_binomial_P</code>	277	<code>gsl_combination_k</code>	126

<code>gsl_combination_memcpy</code>	126	<code>gsl_complex_polar</code>	27
<code>gsl_combination_n</code>	126	<code>gsl_complex_poly_complex_eval</code>	35
<code>gsl_combination_next</code>	126	<code>gsl_complex_pow</code>	30
<code>gsl_combination_prev</code>	126	<code>gsl_complex_pow_real</code>	30
<code>gsl_combination_valid</code>	126	<code>gsl_complex_rect</code>	27
<code>gsl_complex_abs</code>	28	<code>gsl_complex_sec</code>	31
<code>gsl_complex_abs2</code>	28	<code>gsl_complex_sech</code>	32
<code>gsl_complex_add</code>	28	<code>gsl_complex_sin</code>	30
<code>gsl_complex_add_imag</code>	29	<code>gsl_complex_sinh</code>	32
<code>gsl_complex_add_real</code>	28	<code>gsl_complex_sqrt</code>	30
<code>gsl_complex_arccos</code>	31	<code>gsl_complex_sqrt_real</code>	30
<code>gsl_complex_arccos_real</code>	31	<code>gsl_complex_sub</code>	28
<code>gsl_complex_arccosh</code>	33	<code>gsl_complex_sub_imag</code>	29
<code>gsl_complex_arccosh_real</code>	33	<code>gsl_complex_sub_real</code>	29
<code>gsl_complex_arccot</code>	32	<code>gsl_complex_tan</code>	30
<code>gsl_complex_arccoth</code>	33	<code>gsl_complex_tanh</code>	32
<code>gsl_complex_arccsc</code>	32	<code>gsl_deriv_backward</code>	367
<code>gsl_complex_arccsc_real</code>	32	<code>gsl_deriv_central</code>	367
<code>gsl_complex_arccsch</code>	33	<code>gsl_deriv_forward</code>	367
<code>gsl_complex_arcsec</code>	31	<code>gsl_dht_alloc</code>	388
<code>gsl_complex_arcsec_real</code>	31	<code>gsl_dht_apply</code>	388
<code>gsl_complex_arcsech</code>	33	<code>gsl_dht_free</code>	388
<code>gsl_complex_arcsin</code>	31	<code>gsl_dht_init</code>	388
<code>gsl_complex_arcsin_real</code>	31	<code>gsl_dht_k_sample</code>	388
<code>gsl_complex_arcsinh</code>	33	<code>gsl_dht_new</code>	388
<code>gsl_complex_arctan</code>	31	<code>gsl_dht_x_sample</code>	388
<code>gsl_complex_arctanh</code>	33	<code>gsl_eigen_gen</code>	178
<code>gsl_complex_arctanh_real</code>	33	<code>gsl_eigen_gen_alloc</code>	178
<code>gsl_complex_arg</code>	28	<code>gsl_eigen_gen_free</code>	178
<code>gsl_complex_conjugate</code>	29	<code>gsl_eigen_gen_params</code>	178
<code>gsl_complex_cos</code>	30	<code>gsl_eigen_gen_QZ</code>	178
<code>gsl_complex_cosh</code>	32	<code>gsl_eigen_genherm</code>	176
<code>gsl_complex_cot</code>	31	<code>gsl_eigen_genherm_alloc</code>	176
<code>gsl_complex_coth</code>	32	<code>gsl_eigen_genherm_free</code>	176
<code>gsl_complex_csc</code>	31	<code>gsl_eigen_genhermv</code>	177
<code>gsl_complex_csch</code>	32	<code>gsl_eigen_genhermv_alloc</code>	176
<code>gsl_complex_div</code>	28	<code>gsl_eigen_genhermv_free</code>	176
<code>gsl_complex_div_imag</code>	29	<code>gsl_eigen_genhermv_sort</code>	180
<code>gsl_complex_div_real</code>	29	<code>gsl_eigen_gensymm</code>	175
<code>gsl_complex_exp</code>	30	<code>gsl_eigen_gensymm_alloc</code>	175
<code>gsl_complex_inverse</code>	29	<code>gsl_eigen_gensymm_free</code>	175
<code>gsl_complex_log</code>	30	<code>gsl_eigen_gensymmv</code>	176
<code>gsl_complex_log_b</code>	30	<code>gsl_eigen_gensymmv_alloc</code>	175
<code>gsl_complex_log10</code>	30	<code>gsl_eigen_gensymmv_free</code>	175
<code>gsl_complex_logabs</code>	28	<code>gsl_eigen_gensymmv_sort</code>	180
<code>gsl_complex_mul</code>	28	<code>gsl_eigen_genv</code>	179
<code>gsl_complex_mul_imag</code>	29	<code>gsl_eigen_genv_alloc</code>	178
<code>gsl_complex_mul_real</code>	29	<code>gsl_eigen_genv_free</code>	179
<code>gsl_complex_negative</code>	29	<code>gsl_eigen_genv_QZ</code>	179

<code>gsl_eigen_genv_sort</code>	180	<code>gsl_fft_complex_wavetable_free</code> ..	191
<code>gsl_eigen_herm</code>	172	<code>gsl_fft_complex_workspace_alloc</code>	
<code>gsl_eigen_herm_alloc</code>	172	191
<code>gsl_eigen_herm_free</code>	172	<code>gsl_fft_complex_workspace_free</code> ..	191
<code>gsl_eigen_hermv</code>	172	<code>gsl_fft_halfcomplex_radix2_backward</code>	
<code>gsl_eigen_hermv_alloc</code>	172	196
<code>gsl_eigen_hermv_free</code>	172	<code>gsl_fft_halfcomplex_radix2_inverse</code>	
<code>gsl_eigen_hermv_sort</code>	179	196
<code>gsl_eigen_nonsymm</code>	174	<code>gsl_fft_halfcomplex_radix2_unpack</code>	
<code>gsl_eigen_nonsymm_alloc</code>	173	196
<code>gsl_eigen_nonsymm_free</code>	173	<code>gsl_fft_halfcomplex_transform</code> ..	199
<code>gsl_eigen_nonsymm_params</code>	173	<code>gsl_fft_halfcomplex_unpack</code>	199
<code>gsl_eigen_nonsymm_Z</code>	174	<code>gsl_fft_halfcomplex_wavetable_alloc</code>	
<code>gsl_eigen_nonsymmv</code>	174	198
<code>gsl_eigen_nonsymmv_alloc</code>	174	<code>gsl_fft_halfcomplex_wavetable_free</code>	
<code>gsl_eigen_nonsymmv_free</code>	174	198
<code>gsl_eigen_nonsymmv_sort</code>	180	<code>gsl_fft_real_radix2_transform</code> ..	195
<code>gsl_eigen_nonsymmv_Z</code>	174	<code>gsl_fft_real_transform</code>	199
<code>gsl_eigen_symm</code>	171	<code>gsl_fft_real_unpack</code>	199
<code>gsl_eigen_symm_alloc</code>	171	<code>gsl_fft_real_wavetable_alloc</code>	198
<code>gsl_eigen_symm_free</code>	171	<code>gsl_fft_real_wavetable_free</code>	198
<code>gsl_eigen_symmv</code>	172	<code>gsl_fft_real_workspace_alloc</code>	198
<code>gsl_eigen_symmv_alloc</code>	171	<code>gsl_fft_real_workspace_free</code>	198
<code>gsl_eigen_symmv_free</code>	171	<code>gsl_finite</code>	22
<code>gsl_eigen_symmv_sort</code>	179	<code>gsl_fit_linear</code>	444
<code>GSL_ERROR</code>	18	<code>gsl_fit_linear_est</code>	445
<code>GSL_ERROR_VAL</code>	18	<code>gsl_fit_mul</code>	445
<code>gsl_expm1</code>	23	<code>gsl_fit_mul_est</code>	445
<code>gsl_fcmp</code>	26	<code>gsl_fit_wlinear</code>	444
<code>gsl_fft_complex_backward</code>	192	<code>gsl_fit_wmul</code>	445
<code>gsl_fft_complex_forward</code>	192	<code>gsl_frexp</code>	23
<code>gsl_fft_complex_inverse</code>	192	<code>gsl_heapsort</code>	131
<code>gsl_fft_complex_radix2_backward</code>		<code>gsl_heapsort_index</code>	132
.....	187	<code>gsl_histogram_accumulate</code>	302
<code>gsl_fft_complex_radix2_dif_backward</code>		<code>gsl_histogram_add</code>	303
.....	188	<code>gsl_histogram_alloc</code>	300
<code>gsl_fft_complex_radix2_dif_forward</code>		<code>gsl_histogram_bins</code>	302
.....	188	<code>gsl_histogram_clone</code>	301
<code>gsl_fft_complex_radix2_dif_inverse</code>		<code>gsl_histogram_div</code>	304
.....	188	<code>gsl_histogram_equal_bins_p</code>	303
<code>gsl_fft_complex_radix2_dif_</code>		<code>gsl_histogram_find</code>	302
<code>transform</code>	188	<code>gsl_histogram_fprintf</code>	305
<code>gsl_fft_complex_radix2_forward</code> ..	187	<code>gsl_histogram_fread</code>	304
<code>gsl_fft_complex_radix2_inverse</code> ..	187	<code>gsl_histogram_free</code>	301
<code>gsl_fft_complex_radix2_transform</code>		<code>gsl_histogram_fscanf</code>	305
.....	187	<code>gsl_histogram_fwrite</code>	304
<code>gsl_fft_complex_transform</code>	192	<code>gsl_histogram_get</code>	302
<code>gsl_fft_complex_wavetable_alloc</code>		<code>gsl_histogram_get_range</code>	302
.....	190	<code>gsl_histogram_increment</code>	301

<code>gsl_histogram_max</code>	302	<code>gsl_histogram2d_pdf_init</code>	316
<code>gsl_histogram_max_bin</code>	303	<code>gsl_histogram2d_pdf_sample</code>	316
<code>gsl_histogram_max_val</code>	303	<code>gsl_histogram2d_reset</code>	312
<code>gsl_histogram_mean</code>	303	<code>gsl_histogram2d_scale</code>	314
<code>gsl_histogram_memcpy</code>	301	<code>gsl_histogram2d_set_ranges</code>	310
<code>gsl_histogram_min</code>	302	<code>gsl_histogram2d_set_ranges_uniform</code>	310
<code>gsl_histogram_min_bin</code>	303	<code>gsl_histogram2d_shift</code>	314
<code>gsl_histogram_min_val</code>	303	<code>gsl_histogram2d_sub</code>	313
<code>gsl_histogram_mul</code>	304	<code>gsl_histogram2d_sum</code>	313
<code>gsl_histogram_pdf_alloc</code>	306	<code>gsl_histogram2d_xmax</code>	312
<code>gsl_histogram_pdf_free</code>	306	<code>gsl_histogram2d_xmean</code>	313
<code>gsl_histogram_pdf_init</code>	306	<code>gsl_histogram2d_xmin</code>	312
<code>gsl_histogram_pdf_sample</code>	307	<code>gsl_histogram2d_xsigma</code>	313
<code>gsl_histogram_reset</code>	302	<code>gsl_histogram2d_ymax</code>	312
<code>gsl_histogram_scale</code>	304	<code>gsl_histogram2d_ymean</code>	313
<code>gsl_histogram_set_ranges</code>	300	<code>gsl_histogram2d_ymin</code>	312
<code>gsl_histogram_set_ranges_uniform</code>	301	<code>gsl_histogram2d_ysigma</code>	313
<code>gsl_histogram_shift</code>	304	<code>gsl_hypot</code>	23
<code>gsl_histogram_sigma</code>	303	<code>gsl_hypot3</code>	23
<code>gsl_histogram_sub</code>	304	<code>gsl_ieee_env_setup</code>	489
<code>gsl_histogram_sum</code>	303	<code>gsl_ieee_fprintf_double</code>	488
<code>gsl_histogram2d_accumulate</code>	311	<code>gsl_ieee_fprintf_float</code>	488
<code>gsl_histogram2d_add</code>	313	<code>gsl_ieee_printf_double</code>	488
<code>gsl_histogram2d_alloc</code>	310	<code>gsl_ieee_printf_float</code>	488
<code>gsl_histogram2d_clone</code>	310	<code>GSL_IMAG</code>	27
<code>gsl_histogram2d_cov</code>	313	<code>gsl_integration_qag</code>	207
<code>gsl_histogram2d_div</code>	314	<code>gsl_integration_qagi</code>	209
<code>gsl_histogram2d_equal_bins_p</code>	313	<code>gsl_integration_qagil</code>	210
<code>gsl_histogram2d_find</code>	312	<code>gsl_integration_qagiu</code>	209
<code>gsl_histogram2d_fprintf</code>	314	<code>gsl_integration_qagp</code>	209
<code>gsl_histogram2d_fread</code>	314	<code>gsl_integration_qags</code>	208
<code>gsl_histogram2d_free</code>	310	<code>gsl_integration_qawc</code>	210
<code>gsl_histogram2d_fscanf</code>	315	<code>gsl_integration_qawf</code>	213
<code>gsl_histogram2d_fwrite</code>	314	<code>gsl_integration_qawo</code>	213
<code>gsl_histogram2d_get</code>	311	<code>gsl_integration_qawo_table_alloc</code>	212
<code>gsl_histogram2d_get_xrange</code>	311	<code>gsl_integration_qawo_table_free</code>	213
<code>gsl_histogram2d_get_yrange</code>	311	<code>gsl_integration_qawo_table_set</code> ..	212
<code>gsl_histogram2d_increment</code>	311	<code>gsl_integration_qawo_table_set_</code> length.....	213
<code>gsl_histogram2d_max_bin</code>	312	<code>gsl_integration_qaws</code>	211
<code>gsl_histogram2d_max_val</code>	312	<code>gsl_integration_qaws_table_alloc</code>	211
<code>gsl_histogram2d_memcpy</code>	310	<code>gsl_integration_qaws_table_free</code>	211
<code>gsl_histogram2d_min_bin</code>	312	<code>gsl_integration_qaws_table_set</code> ..	211
<code>gsl_histogram2d_min_val</code>	312	<code>gsl_integration_qng</code>	207
<code>gsl_histogram2d_mul</code>	314		
<code>gsl_histogram2d_nx</code>	312		
<code>gsl_histogram2d_ny</code>	312		
<code>gsl_histogram2d_pdf_alloc</code>	316		
<code>gsl_histogram2d_pdf_free</code>	316		

<code>gsl_integration_workspace_alloc</code>	207	<code>gsl_linalg_complex_householder_hv</code>	166
<code>gsl_integration_workspace_free</code>	207	<code>gsl_linalg_complex_householder_mh</code>	165
<code>gsl_interp_accel_alloc</code>	361	<code>gsl_linalg_complex_householder_transform</code>	165
<code>gsl_interp_accel_find</code>	361	<code>gsl_linalg_complex_LU_decomp</code>	153
<code>gsl_interp_accel_free</code>	361	<code>gsl_linalg_complex_LU_det</code>	154
<code>gsl_interp_akima</code>	360	<code>gsl_linalg_complex_LU_invert</code>	154
<code>gsl_interp_akima_periodic</code>	360	<code>gsl_linalg_complex_LU_lndet</code>	154
<code>gsl_interp_alloc</code>	359	<code>gsl_linalg_complex_LU_refine</code>	154
<code>gsl_interp_bsearch</code>	361	<code>gsl_linalg_complex_LU_sgndet</code>	155
<code>gsl_interp_cspline</code>	360	<code>gsl_linalg_complex_LU_solve</code>	153
<code>gsl_interp_cspline_periodic</code>	360	<code>gsl_linalg_complex_LU_svx</code>	154
<code>gsl_interp_eval</code>	361	<code>gsl_linalg_hermtdecomp</code>	162
<code>gsl_interp_eval_deriv</code>	361	<code>gsl_linalg_hermtdeunpack</code>	162
<code>gsl_interp_eval_deriv_e</code>	361	<code>gsl_linalg_hermtdeunpack_T</code>	162
<code>gsl_interp_eval_deriv2</code>	362	<code>gsl_linalg_hessenberg_decomp</code>	163
<code>gsl_interp_eval_deriv2_e</code>	362	<code>gsl_linalg_hessenberg_set_zero</code>	163
<code>gsl_interp_eval_e</code>	361	<code>gsl_linalg_hessenberg_unpack</code>	163
<code>gsl_interp_eval_integ</code>	362	<code>gsl_linalg_hessenberg_unpack_accum</code>	163
<code>gsl_interp_eval_integ_e</code>	362	<code>gsl_linalg_hesstri_decomp</code>	164
<code>gsl_interp_free</code>	359	<code>gsl_linalg_HH_solve</code>	166
<code>gsl_interp_init</code>	359	<code>gsl_linalg_HH_svx</code>	166
<code>gsl_interp_linear</code>	360	<code>gsl_linalg_householder_hm</code>	165
<code>gsl_interp_min_size</code>	360	<code>gsl_linalg_householder_hv</code>	166
<code>gsl_interp_name</code>	360	<code>gsl_linalg_householder_mh</code>	165
<code>gsl_interp_polynomial</code>	360	<code>gsl_linalg_householder_transform</code>	165
<code>GSL_IS_EVEN</code>	24	<code>gsl_linalg_LU_decomp</code>	153
<code>GSL_IS_ODD</code>	24	<code>gsl_linalg_LU_det</code>	154
<code>gsl_isinf</code>	22	<code>gsl_linalg_LU_invert</code>	154
<code>gsl_isnan</code>	22	<code>gsl_linalg_LU_lndet</code>	154
<code>gsl_ldexp</code>	23	<code>gsl_linalg_LU_refine</code>	154
<code>gsl_linalg_balance_matrix</code>	167	<code>gsl_linalg_LU_sgndet</code>	155
<code>gsl_linalg_bidiag_decomp</code>	164	<code>gsl_linalg_LU_solve</code>	153
<code>gsl_linalg_bidiag_unpack</code>	164	<code>gsl_linalg_LU_svx</code>	154
<code>gsl_linalg_bidiag_unpack_B</code>	165	<code>gsl_linalg_QR_decomp</code>	155
<code>gsl_linalg_bidiag_unpack2</code>	165	<code>gsl_linalg_QR_lssolve</code>	156
<code>gsl_linalg_cholesky_decomp</code>	160	<code>gsl_linalg_QR_QRsolve</code>	157
<code>gsl_linalg_cholesky_invert</code>	161	<code>gsl_linalg_QR_QTmat</code>	156
<code>gsl_linalg_cholesky_solve</code>	161	<code>gsl_linalg_QR_QTvec</code>	156
<code>gsl_linalg_cholesky_svx</code>	161	<code>gsl_linalg_QR_Qvec</code>	156
<code>gsl_linalg_complex_cholesky_decomp</code>	160	<code>gsl_linalg_QR_Rsolve</code>	156
<code>gsl_linalg_complex_cholesky_solve</code>	161	<code>gsl_linalg_QR_Rsvx</code>	156
<code>gsl_linalg_complex_cholesky_svx</code>	161	<code>gsl_linalg_QR_solve</code>	155
<code>gsl_linalg_complex_householder_hm</code>	165	<code>gsl_linalg_QR_svx</code>	155
		<code>gsl_linalg_QR_unpack</code>	156

<code>gsl_linalg_QR_update</code>	157	<code>gsl_matrix_fscanf</code>	104
<code>gsl_linalg_QRPT_decomp</code>	157	<code>gsl_matrix_fwrite</code>	104
<code>gsl_linalg_QRPT_decomp2</code>	158	<code>gsl_matrix_get</code>	103
<code>gsl_linalg_QRPT_QRsolve</code>	158	<code>gsl_matrix_get_col</code>	109
<code>gsl_linalg_QRPT_Rsolve</code>	158	<code>gsl_matrix_get_row</code>	109
<code>gsl_linalg_QRPT_Rsvx</code>	158	<code>gsl_matrix_isneg</code>	111
<code>gsl_linalg_QRPT_solve</code>	158	<code>gsl_matrix_isnonneg</code>	111
<code>gsl_linalg_QRPT_svx</code>	158	<code>gsl_matrix_isnull</code>	111
<code>gsl_linalg_QRPT_update</code>	158	<code>gsl_matrix_ispos</code>	111
<code>gsl_linalg_R_solve</code>	157	<code>gsl_matrix_max</code>	111
<code>gsl_linalg_R_svx</code>	157	<code>gsl_matrix_max_index</code>	111
<code>gsl_linalg_solve_cyc_tridiag</code>	167	<code>gsl_matrix_memcpy</code>	108
<code>gsl_linalg_solve_symm_cyc_tridiag</code>	167	<code>gsl_matrix_min</code>	111
<code>gsl_linalg_solve_symm_tridiag</code> ...	166	<code>gsl_matrix_min_index</code>	111
<code>gsl_linalg_solve_tridiag</code>	166	<code>gsl_matrix_minmax</code>	111
<code>gsl_linalg_SV_decomp</code>	159	<code>gsl_matrix_minmax_index</code>	111
<code>gsl_linalg_SV_decomp_jacobi</code>	160	<code>gsl_matrix_mul_elements</code>	110
<code>gsl_linalg_SV_decomp_mod</code>	159	<code>gsl_matrix_ptr</code>	103
<code>gsl_linalg_SV_solve</code>	160	<code>gsl_matrix_row</code>	107
<code>gsl_linalg_symmtd_decomp</code>	161	<code>gsl_matrix_scale</code>	110
<code>gsl_linalg_symmtd_unpack</code>	162	<code>gsl_matrix_set</code>	103
<code>gsl_linalg_symmtd_unpack_T</code>	162	<code>gsl_matrix_set_all</code>	103
<code>gsl_log1p</code>	23	<code>gsl_matrix_set_col</code>	109
<code>gsl_matrix_add</code>	110	<code>gsl_matrix_set_identity</code>	103
<code>gsl_matrix_add_constant</code>	110	<code>gsl_matrix_set_row</code>	109
<code>gsl_matrix_alloc</code>	102	<code>gsl_matrix_set_zero</code>	103
<code>gsl_matrix_calloc</code>	102	<code>gsl_matrix_sub</code>	110
<code>gsl_matrix_column</code>	107	<code>gsl_matrix_subcolumn</code>	107
<code>gsl_matrix_const_column</code>	107	<code>gsl_matrix_subdiagonal</code>	108
<code>gsl_matrix_const_diagonal</code>	108	<code>gsl_matrix_submatrix</code>	105
<code>gsl_matrix_const_ptr</code>	103	<code>gsl_matrix_subrow</code>	107
<code>gsl_matrix_const_row</code>	107	<code>gsl_matrix_superdiagonal</code>	108
<code>gsl_matrix_const_subcolumn</code>	107	<code>gsl_matrix_swap</code>	108
<code>gsl_matrix_const_subdiagonal</code>	108	<code>gsl_matrix_swap_columns</code>	109
<code>gsl_matrix_const_submatrix</code>	105	<code>gsl_matrix_swap_rowcol</code>	109
<code>gsl_matrix_const_subrow</code>	107	<code>gsl_matrix_swap_rows</code>	109
<code>gsl_matrix_const_superdiagonal</code> ..	108	<code>gsl_matrix_transpose</code>	110
<code>gsl_matrix_const_view_array</code>	105	<code>gsl_matrix_transpose_memcpy</code>	110
<code>gsl_matrix_const_view_array_with_</code> <code>tda</code>	105	<code>gsl_matrix_view_array</code>	105
<code>gsl_matrix_const_view_vector</code>	106	<code>gsl_matrix_view_array_with_tda</code> ..	105
<code>gsl_matrix_const_view_vector_with_</code> <code>tda</code>	106	<code>gsl_matrix_view_vector</code>	106
<code>gsl_matrix_diagonal</code>	108	<code>gsl_matrix_view_vector_with_tda</code>	106
<code>gsl_matrix_div_elements</code>	110	<code>GSL_MAX</code>	25
<code>gsl_matrix_fprintf</code>	104	<code>GSL_MAX_DBL</code>	25
<code>gsl_matrix_fread</code>	104	<code>GSL_MAX_INT</code>	25
<code>gsl_matrix_free</code>	102	<code>GSL_MAX_LDBL</code>	25
		<code>GSL_MIN</code>	25
		<code>GSL_MIN_DBL</code>	25

<code>gsl_min_fminimizer_alloc</code>	406	<code>gsl_multifit_linear_alloc</code>	446
<code>gsl_min_fminimizer_brent</code>	409	<code>gsl_multifit_linear_est</code>	447
<code>gsl_min_fminimizer_f_lower</code>	408	<code>gsl_multifit_linear_free</code>	446
<code>gsl_min_fminimizer_f_minimum</code>	408	<code>gsl_multifit_linear_residuals</code> ...	447
<code>gsl_min_fminimizer_f_upper</code>	408	<code>gsl_multifit_linear_svd</code>	446
<code>gsl_min_fminimizer_free</code>	407	<code>gsl_multifit_test_delta</code>	459
<code>gsl_min_fminimizer_goldensection</code>	409	<code>gsl_multifit_test_gradient</code>	459
<code>gsl_min_fminimizer_iterate</code>	407	<code>gsl_multifit_wlinear</code>	447
<code>gsl_min_fminimizer_name</code>	407	<code>gsl_multifit_wlinear_svd</code>	447
<code>gsl_min_fminimizer_set</code>	407	<code>gsl_multimin_fdfminimizer_alloc</code>	430
<code>gsl_min_fminimizer_set_with_values</code>	407	<code>gsl_multimin_fdfminimizer_</code> <code>conjugate_fr</code>	435
<code>gsl_min_fminimizer_x_lower</code>	408	<code>gsl_multimin_fdfminimizer_</code> <code>conjugate_pr</code>	435
<code>gsl_min_fminimizer_x_minimum</code>	408	<code>gsl_multimin_fdfminimizer_free</code> ..	431
<code>gsl_min_fminimizer_x_upper</code>	408	<code>gsl_multimin_fdfminimizer_gradient</code>	434
<code>GSL_MIN_INT</code>	25	<code>gsl_multimin_fdfminimizer_iterate</code>	433
<code>GSL_MIN_LDBL</code>	25	<code>gsl_multimin_fdfminimizer_minimum</code>	434
<code>gsl_min_test_interval</code>	408	<code>gsl_multimin_fdfminimizer_name</code> ..	431
<code>gsl_monte_miser_alloc</code>	328	<code>gsl_multimin_fdfminimizer_restart</code>	434
<code>gsl_monte_miser_free</code>	329	<code>gsl_multimin_fdfminimizer_set</code> ...	430
<code>gsl_monte_miser_init</code>	328	<code>gsl_multimin_fdfminimizer_steepest_</code> <code>descent</code>	435
<code>gsl_monte_miser_integrate</code>	329	<code>gsl_multimin_fdfminimizer_vector_</code> <code>bfgs</code>	435
<code>gsl_monte_plain_alloc</code>	327	<code>gsl_multimin_fdfminimizer_vector_</code> <code>bfgs2</code>	435
<code>gsl_monte_plain_free</code>	327	<code>gsl_multimin_fdfminimizer_x</code>	434
<code>gsl_monte_plain_init</code>	327	<code>gsl_multimin_fminimizer_alloc</code> ...	430
<code>gsl_monte_plain_integrate</code>	327	<code>gsl_multimin_fminimizer_free</code>	431
<code>gsl_monte_vegas_alloc</code>	331	<code>gsl_multimin_fminimizer_iterate</code>	433
<code>gsl_monte_vegas_free</code>	331	<code>gsl_multimin_fminimizer_minimum</code>	434
<code>gsl_monte_vegas_init</code>	331	<code>gsl_multimin_fminimizer_name</code>	431
<code>gsl_monte_vegas_integrate</code>	331	<code>gsl_multimin_fminimizer_nmsimplex</code>	436
<code>gsl_multifit_covar</code>	461	<code>gsl_multimin_fminimizer_nmsimplex2</code>	436
<code>gsl_multifit_fdfsolver_alloc</code>	456	<code>gsl_multimin_fminimizer_set</code>	430
<code>gsl_multifit_fdfsolver_free</code>	456	<code>gsl_multimin_fminimizer_size</code>	434
<code>gsl_multifit_fdfsolver_iterate</code> ..	458	<code>gsl_multimin_fminimizer_x</code>	434
<code>gsl_multifit_fdfsolver_lmder</code>	460	<code>gsl_multimin_test_gradient</code>	434
<code>gsl_multifit_fdfsolver_lmsder</code> ...	460	<code>gsl_multimin_test_size</code>	434
<code>gsl_multifit_fdfsolver_name</code>	456		
<code>gsl_multifit_fdfsolver_position</code>	458		
<code>gsl_multifit_fdfsolver_set</code>	456		
<code>gsl_multifit_fsolver_alloc</code>	456		
<code>gsl_multifit_fsolver_free</code>	456		
<code>gsl_multifit_fsolver_iterate</code>	458		
<code>gsl_multifit_fsolver_name</code>	456		
<code>gsl_multifit_fsolver_position</code> ...	458		
<code>gsl_multifit_fsolver_set</code>	456		
<code>gsl_multifit_gradient</code>	459		
<code>gsl_multifit_linear</code>	446		

<code>gsl_multiroot_fdfsolver_alloc</code> ...	414	<code>gsl_odeiv_evolve_reset</code>	352
<code>gsl_multiroot_fdfsolver_dx</code>	418	<code>gsl_odeiv_step_alloc</code>	348
<code>gsl_multiroot_fdfsolver_f</code>	418	<code>gsl_odeiv_step_apply</code>	348
<code>gsl_multiroot_fdfsolver_free</code>	415	<code>gsl_odeiv_step_bsimp</code>	349
<code>gsl_multiroot_fdfsolver_gnewton</code>	421	<code>gsl_odeiv_step_free</code>	348
<code>gsl_multiroot_fdfsolver_hybridj</code>	421	<code>gsl_odeiv_step_gear1</code>	349
<code>gsl_multiroot_fdfsolver_hybridjs</code>	420	<code>gsl_odeiv_step_gear2</code>	349
<code>gsl_multiroot_fdfsolver_iterate</code>	418	<code>gsl_odeiv_step_name</code>	348
<code>gsl_multiroot_fdfsolver_name</code>	415	<code>gsl_odeiv_step_order</code>	348
<code>gsl_multiroot_fdfsolver_newton</code> ..	421	<code>gsl_odeiv_step_reset</code>	348
<code>gsl_multiroot_fdfsolver_root</code>	418	<code>gsl_odeiv_step_rk2</code>	349
<code>gsl_multiroot_fdfsolver_set</code>	414	<code>gsl_odeiv_step_rk2imp</code>	349
<code>gsl_multiroot_fsolver_alloc</code>	414	<code>gsl_odeiv_step_rk4</code>	349
<code>gsl_multiroot_fsolver_broyden</code> ...	422	<code>gsl_odeiv_step_rk4imp</code>	349
<code>gsl_multiroot_fsolver_dnewton</code> ...	422	<code>gsl_odeiv_step_rk8pd</code>	349
<code>gsl_multiroot_fsolver_dx</code>	418	<code>gsl_odeiv_step_rkck</code>	349
<code>gsl_multiroot_fsolver_f</code>	418	<code>gsl_odeiv_step_rkf45</code>	349
<code>gsl_multiroot_fsolver_free</code>	415	<code>gsl_permutation_alloc</code>	117
<code>gsl_multiroot_fsolver_hybrid</code>	421	<code>gsl_permutation_calloc</code>	117
<code>gsl_multiroot_fsolver_hybrids</code> ...	421	<code>gsl_permutation_canonical_cycles</code>	122
<code>gsl_multiroot_fsolver_iterate</code> ...	418	<code>gsl_permutation_canonical_to_linear</code>	121
<code>gsl_multiroot_fsolver_name</code>	415	<code>gsl_permutation_data</code>	118
<code>gsl_multiroot_fsolver_root</code>	418	<code>gsl_permutation_fprintf</code>	120
<code>gsl_multiroot_fsolver_set</code>	414	<code>gsl_permutation_fread</code>	120
<code>gsl_multiroot_test_delta</code>	419	<code>gsl_permutation_free</code>	117
<code>gsl_multiroot_test_residual</code>	419	<code>gsl_permutation_fscanf</code>	120
<code>gsl_ntuple_bookdata</code>	320	<code>gsl_permutation_fwrite</code>	120
<code>gsl_ntuple_close</code>	320	<code>gsl_permutation_get</code>	118
<code>gsl_ntuple_create</code>	319	<code>gsl_permutation_init</code>	117
<code>gsl_ntuple_open</code>	319	<code>gsl_permutation_inverse</code>	118
<code>gsl_ntuple_project</code>	321	<code>gsl_permutation_inversions</code>	121
<code>gsl_ntuple_read</code>	320	<code>gsl_permutation_linear_cycles</code> ...	122
<code>gsl_ntuple_write</code>	320	<code>gsl_permutation_linear_to_canonical</code>	121
<code>gsl_odeiv_control_alloc</code>	351	<code>gsl_permutation_memcpy</code>	118
<code>gsl_odeiv_control_free</code>	351	<code>gsl_permutation_mul</code>	119
<code>gsl_odeiv_control_hadjust</code>	351	<code>gsl_permutation_next</code>	118
<code>gsl_odeiv_control_init</code>	351	<code>gsl_permutation_prev</code>	119
<code>gsl_odeiv_control_name</code>	351	<code>gsl_permutation_reverse</code>	118
<code>gsl_odeiv_control_scaled_new</code>	351	<code>gsl_permutation_size</code>	118
<code>gsl_odeiv_control_standard_new</code> ..	350	<code>gsl_permutation_swap</code>	118
<code>gsl_odeiv_control_y_new</code>	350	<code>gsl_permutation_valid</code>	118
<code>gsl_odeiv_control_yp_new</code>	350	<code>gsl_permute</code>	119
<code>gsl_odeiv_evolve_alloc</code>	352	<code>gsl_permute_inverse</code>	119
<code>gsl_odeiv_evolve_apply</code>	352	<code>gsl_permute_vector</code>	119
<code>gsl_odeiv_evolve_free</code>	352	<code>gsl_permute_vector_inverse</code>	119
		<code>gsl_poly_complex_eval</code>	35

<code>gsl_poly_complex_solve</code>	38	<code>gsl_ran_dir_2d_trig_method</code>	266
<code>gsl_poly_complex_solve_cubic</code>	37	<code>gsl_ran_dir_3d</code>	266
<code>gsl_poly_complex_solve_quadratic</code>	36	<code>gsl_ran_dir_nd</code>	266
<code>gsl_poly_complex_workspace_alloc</code>	37	<code>gsl_ran_dirichlet</code>	270
<code>gsl_poly_complex_workspace_free</code> ..	37	<code>gsl_ran_dirichlet_lnpdf</code>	270
<code>gsl_poly_dd_eval</code>	35	<code>gsl_ran_dirichlet_pdf</code>	270
<code>gsl_poly_dd_init</code>	35	<code>gsl_ran_discrete</code>	272
<code>gsl_poly_dd_taylor</code>	36	<code>gsl_ran_discrete_free</code>	272
<code>gsl_poly_eval</code>	35	<code>gsl_ran_discrete_pdf</code>	272
<code>gsl_poly_solve_cubic</code>	37	<code>gsl_ran_discrete_preproc</code>	271
<code>gsl_poly_solve_quadratic</code>	36	<code>gsl_ran_exponential</code>	246
<code>gsl_pow_2</code>	24	<code>gsl_ran_exponential_pdf</code>	246
<code>gsl_pow_3</code>	24	<code>gsl_ran_exppow</code>	248
<code>gsl_pow_4</code>	24	<code>gsl_ran_exppow_pdf</code>	248
<code>gsl_pow_5</code>	24	<code>gsl_ran_fdist</code>	260
<code>gsl_pow_6</code>	24	<code>gsl_ran_fdist_pdf</code>	260
<code>gsl_pow_7</code>	24	<code>gsl_ran_flat</code>	257
<code>gsl_pow_8</code>	24	<code>gsl_ran_flat_pdf</code>	257
<code>gsl_pow_9</code>	24	<code>gsl_ran_gamma</code>	255
<code>gsl_pow_int</code>	24	<code>gsl_ran_gamma_knuth</code>	255
<code>gsl_qrng_alloc</code>	235	<code>gsl_ran_gamma_pdf</code>	255
<code>gsl_qrng_clone</code>	236	<code>gsl_ran_gaussian</code>	241
<code>gsl_qrng_free</code>	235	<code>gsl_ran_gaussian_pdf</code>	241
<code>gsl_qrng_get</code>	235	<code>gsl_ran_gaussian_ratio_method</code> ..	241
<code>gsl_qrng_haltom</code>	236	<code>gsl_ran_gaussian_tail</code>	243
<code>gsl_qrng_init</code>	235	<code>gsl_ran_gaussian_tail_pdf</code>	243
<code>gsl_qrng_memcpy</code>	236	<code>gsl_ran_gaussian_ziggurat</code>	241
<code>gsl_qrng_name</code>	236	<code>gsl_ran_geometric</code>	279
<code>gsl_qrng_niederreiter_2</code>	236	<code>gsl_ran_geometric_pdf</code>	279
<code>gsl_qrng_reversehaltom</code>	236	<code>gsl_ran_gumbel1</code>	268
<code>gsl_qrng_size</code>	236	<code>gsl_ran_gumbel1_pdf</code>	268
<code>gsl_qrng_sobol</code>	236	<code>gsl_ran_gumbel2</code>	269
<code>gsl_qrng_state</code>	236	<code>gsl_ran_gumbel2_pdf</code>	269
<code>gsl_ran_bernoulli</code>	274	<code>gsl_ran_hypergeometric</code>	280
<code>gsl_ran_bernoulli_pdf</code>	274	<code>gsl_ran_hypergeometric_pdf</code>	280
<code>gsl_ran_beta</code>	263	<code>gsl_ran_landau</code>	252
<code>gsl_ran_beta_pdf</code>	263	<code>gsl_ran_landau_pdf</code>	252
<code>gsl_ran_binomial</code>	275	<code>gsl_ran_laplace</code>	247
<code>gsl_ran_binomial_pdf</code>	275	<code>gsl_ran_laplace_pdf</code>	247
<code>gsl_ran_bivariate_gaussian</code>	245	<code>gsl_ran_levy</code>	253
<code>gsl_ran_bivariate_gaussian_pdf</code> ..	245	<code>gsl_ran_levy_skew</code>	254
<code>gsl_ran_cauchy</code>	249	<code>gsl_ran_logarithmic</code>	281
<code>gsl_ran_cauchy_pdf</code>	249	<code>gsl_ran_logarithmic_pdf</code>	281
<code>gsl_ran_chisq</code>	259	<code>gsl_ran_logistic</code>	264
<code>gsl_ran_chisq_pdf</code>	259	<code>gsl_ran_logistic_pdf</code>	264
<code>gsl_ran_choose</code>	282	<code>gsl_ran_lognormal</code>	258
<code>gsl_ran_dir_2d</code>	266	<code>gsl_ran_lognormal_pdf</code>	258
		<code>gsl_ran_multinomial</code>	276
		<code>gsl_ran_multinomial_lnpdf</code>	276

<code>gsl_ran_multinomial_pdf</code>	276	<code>gsl_rng_name</code>	220
<code>gsl_ran_negative_binomial</code>	277	<code>gsl_rng_r250</code>	229
<code>gsl_ran_negative_binomial_pdf</code> ...	277	<code>gsl_rng_rand</code>	227
<code>gsl_ran_pareto</code>	265	<code>gsl_rng_rand48</code>	228
<code>gsl_ran_pareto_pdf</code>	265	<code>gsl_rng_random_bsd</code>	227
<code>gsl_ran_pascal</code>	278	<code>gsl_rng_random_glibc2</code>	227
<code>gsl_ran_pascal_pdf</code>	278	<code>gsl_rng_random_libc5</code>	227
<code>gsl_ran_poisson</code>	273	<code>gsl_rng_randu</code>	230
<code>gsl_ran_poisson_pdf</code>	273	<code>gsl_rng_ranf</code>	228
<code>gsl_ran_rayleigh</code>	250	<code>gsl_rng_ranlux</code>	224
<code>gsl_ran_rayleigh_pdf</code>	250	<code>gsl_rng_ranlux389</code>	224
<code>gsl_ran_rayleigh_tail</code>	251	<code>gsl_rng_ranlxd1</code>	224
<code>gsl_ran_rayleigh_tail_pdf</code>	251	<code>gsl_rng_ranlxd2</code>	224
<code>gsl_ran_sample</code>	283	<code>gsl_rng_ranlxs0</code>	224
<code>gsl_ran_shuffle</code>	282	<code>gsl_rng_ranlxs1</code>	224
<code>gsl_ran_tdist</code>	262	<code>gsl_rng_ranlxs2</code>	224
<code>gsl_ran_tdist_pdf</code>	262	<code>gsl_rng_ranmar</code>	229
<code>gsl_ran_ugaussian</code>	242	<code>gsl_rng_set</code>	218
<code>gsl_ran_ugaussian_pdf</code>	242	<code>gsl_rng_size</code>	220
<code>gsl_ran_ugaussian_ratio_method</code> ..	242	<code>gsl_rng_slaterc</code>	230
<code>gsl_ran_ugaussian_tail</code>	244	<code>gsl_rng_state</code>	220
<code>gsl_ran_ugaussian_tail_pdf</code>	244	<code>gsl_rng_taus</code>	225
<code>gsl_ran_weibull</code>	267	<code>gsl_rng_taus2</code>	225
<code>gsl_ran_weibull_pdf</code>	267	<code>gsl_rng_transputer</code>	230
<code>GSL_REAL</code>	27	<code>gsl_rng_tt800</code>	229
<code>gsl_rng_alloc</code>	218	<code>gsl_rng_types_setup</code>	220
<code>gsl_rng_borosh13</code>	231	<code>gsl_rng_uni</code>	230
<code>gsl_rng_clone</code>	222	<code>gsl_rng_uni32</code>	230
<code>gsl_rng_cmrg</code>	225	<code>gsl_rng_uniform</code>	219
<code>gsl_rng_coveyou</code>	231	<code>gsl_rng_uniform_int</code>	219
<code>gsl_rng_env_setup</code>	221	<code>gsl_rng_uniform_pos</code>	219
<code>gsl_rng_fishman18</code>	231	<code>gsl_rng_vax</code>	229
<code>gsl_rng_fishman20</code>	231	<code>gsl_rng_waterman14</code>	231
<code>gsl_rng_fishman2x</code>	231	<code>gsl_rng_zuf</code>	230
<code>gsl_rng_fread</code>	223	<code>gsl_root_fdfsolver_alloc</code>	390
<code>gsl_rng_free</code>	219	<code>gsl_root_fdfsolver_free</code>	391
<code>gsl_rng_fwrite</code>	222	<code>gsl_root_fdfsolver_iterate</code>	394
<code>gsl_rng_get</code>	219	<code>gsl_root_fdfsolver_name</code>	391
<code>gsl_rng_gfsr4</code>	226	<code>gsl_root_fdfsolver_newton</code>	397
<code>gsl_rng_knuthran</code>	231	<code>gsl_root_fdfsolver_root</code>	394
<code>gsl_rng_knuthran2</code>	231	<code>gsl_root_fdfsolver_secant</code>	397
<code>gsl_rng_knuthran2002</code>	231	<code>gsl_root_fdfsolver_set</code>	391
<code>gsl_rng_lecuyer21</code>	231	<code>gsl_root_fdfsolver_steffenson</code> ...	398
<code>gsl_rng_max</code>	220	<code>gsl_root_fsolver_alloc</code>	390
<code>gsl_rng_memcpy</code>	222	<code>gsl_root_fsolver_bisection</code>	396
<code>gsl_rng_min</code>	220	<code>gsl_root_fsolver_brent</code>	396
<code>gsl_rng_minstd</code>	230	<code>gsl_root_fsolver_falsepos</code>	396
<code>gsl_rng_mrg</code>	225	<code>gsl_root_fsolver_free</code>	391
<code>gsl_rng_mt19937</code>	223	<code>gsl_root_fsolver_iterate</code>	394

<code>gsl_root_fsolver_name</code>	391	<code>gsl_sf_bessel_I1</code>	46
<code>gsl_root_fsolver_root</code>	394	<code>gsl_sf_bessel_I1_e</code>	46
<code>gsl_root_fsolver_set</code>	391	<code>gsl_sf_bessel_i1_scaled</code>	50
<code>gsl_root_fsolver_x_lower</code>	394	<code>gsl_sf_bessel_I1_scaled</code>	46
<code>gsl_root_fsolver_x_upper</code>	394	<code>gsl_sf_bessel_i1_scaled_e</code>	50
<code>gsl_root_test_delta</code>	395	<code>gsl_sf_bessel_I1_scaled_e</code>	46
<code>gsl_root_test_interval</code>	395	<code>gsl_sf_bessel_i2_scaled</code>	50
<code>gsl_root_test_residual</code>	395	<code>gsl_sf_bessel_i2_scaled_e</code>	50
<code>GSL_SET_COMPLEX</code>	28	<code>gsl_sf_bessel_il_scaled</code>	50
<code>gsl_set_error_handler</code>	17	<code>gsl_sf_bessel_il_scaled_array</code>	50
<code>gsl_set_error_handler_off</code>	18	<code>gsl_sf_bessel_il_scaled_e</code>	50
<code>GSL_SET_IMAG</code>	28	<code>gsl_sf_bessel_In</code>	46
<code>GSL_SET_REAL</code>	28	<code>gsl_sf_bessel_In_array</code>	46
<code>gsl_sf_airy_Ai</code>	43	<code>gsl_sf_bessel_In_e</code>	46
<code>gsl_sf_airy_Ai_deriv</code>	43	<code>gsl_sf_bessel_In_scaled</code>	46
<code>gsl_sf_airy_Ai_deriv_e</code>	43	<code>gsl_sf_bessel_In_scaled_array</code>	47
<code>gsl_sf_airy_Ai_deriv_scaled</code>	44	<code>gsl_sf_bessel_In_scaled_e</code>	46
<code>gsl_sf_airy_Ai_deriv_scaled_e</code>	44	<code>gsl_sf_bessel_Inu</code>	51
<code>gsl_sf_airy_Ai_e</code>	43	<code>gsl_sf_bessel_Inu_e</code>	51
<code>gsl_sf_airy_Ai_scaled</code>	43	<code>gsl_sf_bessel_Inu_scaled</code>	52
<code>gsl_sf_airy_Ai_scaled_e</code>	43	<code>gsl_sf_bessel_Inu_scaled_e</code>	52
<code>gsl_sf_airy_Bi</code>	43	<code>gsl_sf_bessel_j0</code>	48
<code>gsl_sf_airy_Bi_deriv</code>	44	<code>gsl_sf_bessel_J0</code>	45
<code>gsl_sf_airy_Bi_deriv_e</code>	44	<code>gsl_sf_bessel_j0_e</code>	48
<code>gsl_sf_airy_Bi_deriv_scaled</code>	44	<code>gsl_sf_bessel_J0_e</code>	45
<code>gsl_sf_airy_Bi_deriv_scaled_e</code>	44	<code>gsl_sf_bessel_j1</code>	48
<code>gsl_sf_airy_Bi_e</code>	43	<code>gsl_sf_bessel_J1</code>	45
<code>gsl_sf_airy_Bi_scaled</code>	43	<code>gsl_sf_bessel_j1_e</code>	48
<code>gsl_sf_airy_Bi_scaled_e</code>	43	<code>gsl_sf_bessel_J1_e</code>	45
<code>gsl_sf_airy_zero_Ai</code>	44	<code>gsl_sf_bessel_j2</code>	48
<code>gsl_sf_airy_zero_Ai_deriv</code>	44	<code>gsl_sf_bessel_j2_e</code>	48
<code>gsl_sf_airy_zero_Ai_deriv_e</code>	44	<code>gsl_sf_bessel_jl</code>	48
<code>gsl_sf_airy_zero_Ai_e</code>	44	<code>gsl_sf_bessel_jl_array</code>	48
<code>gsl_sf_airy_zero_Bi</code>	44	<code>gsl_sf_bessel_jl_e</code>	48
<code>gsl_sf_airy_zero_Bi_deriv</code>	44	<code>gsl_sf_bessel_jl_stepped_array</code>	49
<code>gsl_sf_airy_zero_Bi_deriv_e</code>	44	<code>gsl_sf_bessel_Jn</code>	45
<code>gsl_sf_airy_zero_Bi_e</code>	44	<code>gsl_sf_bessel_Jn_array</code>	45
<code>gsl_sf_angle_restrict_pos</code>	85	<code>gsl_sf_bessel_Jn_e</code>	45
<code>gsl_sf_angle_restrict_pos_e</code>	85	<code>gsl_sf_bessel_Jnu</code>	51
<code>gsl_sf_angle_restrict_symm</code>	85	<code>gsl_sf_bessel_Jnu_e</code>	51
<code>gsl_sf_angle_restrict_symm_e</code>	85	<code>gsl_sf_bessel_K0</code>	47
<code>gsl_sf_atanint</code>	66	<code>gsl_sf_bessel_K0_e</code>	47
<code>gsl_sf_atanint_e</code>	66	<code>gsl_sf_bessel_k0_scaled</code>	50
<code>gsl_sf_bessel_I0</code>	46	<code>gsl_sf_bessel_K0_scaled</code>	47
<code>gsl_sf_bessel_I0_e</code>	46	<code>gsl_sf_bessel_k0_scaled_e</code>	50
<code>gsl_sf_bessel_i0_scaled</code>	49	<code>gsl_sf_bessel_K0_scaled_e</code>	47
<code>gsl_sf_bessel_I0_scaled</code>	46	<code>gsl_sf_bessel_K1</code>	47
<code>gsl_sf_bessel_i0_scaled_e</code>	49	<code>gsl_sf_bessel_K1_e</code>	47
<code>gsl_sf_bessel_I0_scaled_e</code>	46	<code>gsl_sf_bessel_k1_scaled</code>	50

<code>gsl_sf_bessel_K1_scaled</code>	47	<code>gsl_sf_Chi</code>	66
<code>gsl_sf_bessel_k1_scaled_e</code>	50	<code>gsl_sf_Chi_e</code>	66
<code>gsl_sf_bessel_K1_scaled_e</code>	47	<code>gsl_sf_choose</code>	70
<code>gsl_sf_bessel_k2_scaled</code>	50	<code>gsl_sf_choose_e</code>	70
<code>gsl_sf_bessel_k2_scaled_e</code>	50	<code>gsl_sf_Ci</code>	66
<code>gsl_sf_bessel_k1_scaled</code>	51	<code>gsl_sf_Ci_e</code>	66
<code>gsl_sf_bessel_k1_scaled_array</code>	51	<code>gsl_sf_clausen</code>	53
<code>gsl_sf_bessel_k1_scaled_e</code>	51	<code>gsl_sf_clausen_e</code>	53
<code>gsl_sf_bessel_Kn</code>	47	<code>gsl_sf_complex_cos_e</code>	84
<code>gsl_sf_bessel_Kn_array</code>	47	<code>gsl_sf_complex_dilog_e</code>	58
<code>gsl_sf_bessel_Kn_e</code>	47	<code>gsl_sf_complex_log_e</code>	79
<code>gsl_sf_bessel_Kn_scaled</code>	48	<code>gsl_sf_complex_logsin_e</code>	84
<code>gsl_sf_bessel_Kn_scaled_array</code>	48	<code>gsl_sf_complex_sin_e</code>	84
<code>gsl_sf_bessel_Kn_scaled_e</code>	48	<code>gsl_sf_conicalP_0</code>	78
<code>gsl_sf_bessel_Knu</code>	52	<code>gsl_sf_conicalP_0_e</code>	78
<code>gsl_sf_bessel_Knu_e</code>	52	<code>gsl_sf_conicalP_1</code>	78
<code>gsl_sf_bessel_Knu_scaled</code>	52	<code>gsl_sf_conicalP_1_e</code>	78
<code>gsl_sf_bessel_Knu_scaled_e</code>	52	<code>gsl_sf_conicalP_cyl_reg</code>	78
<code>gsl_sf_bessel_lnKnu</code>	52	<code>gsl_sf_conicalP_cyl_reg_e</code>	78
<code>gsl_sf_bessel_lnKnu_e</code>	52	<code>gsl_sf_conicalP_half</code>	77
<code>gsl_sf_bessel_sequence_Jnu_e</code>	51	<code>gsl_sf_conicalP_half_e</code>	77
<code>gsl_sf_bessel_y0</code>	49	<code>gsl_sf_conicalP_mhalf</code>	78
<code>gsl_sf_bessel_Y0</code>	45	<code>gsl_sf_conicalP_mhalf_e</code>	78
<code>gsl_sf_bessel_y0_e</code>	49	<code>gsl_sf_conicalP_sph_reg</code>	78
<code>gsl_sf_bessel_Y0_e</code>	45	<code>gsl_sf_conicalP_sph_reg_e</code>	78
<code>gsl_sf_bessel_y1</code>	49	<code>gsl_sf_cos</code>	84
<code>gsl_sf_bessel_Y1</code>	45	<code>gsl_sf_cos_e</code>	84
<code>gsl_sf_bessel_y1_e</code>	49	<code>gsl_sf_cos_err_e</code>	85
<code>gsl_sf_bessel_Y1_e</code>	45	<code>gsl_sf_coulomb_CL_array</code>	55
<code>gsl_sf_bessel_y2</code>	49	<code>gsl_sf_coulomb_CL_e</code>	55
<code>gsl_sf_bessel_y2_e</code>	49	<code>gsl_sf_coulomb_wave_F_array</code>	54
<code>gsl_sf_bessel_y1</code>	49	<code>gsl_sf_coulomb_wave_FG_array</code>	54
<code>gsl_sf_bessel_y1_array</code>	49	<code>gsl_sf_coulomb_wave_FG_e</code>	54
<code>gsl_sf_bessel_y1_e</code>	49	<code>gsl_sf_coulomb_wave_FGp_array</code>	54
<code>gsl_sf_bessel_Yn</code>	45	<code>gsl_sf_coulomb_wave_sphF_array</code> ...	55
<code>gsl_sf_bessel_Yn_array</code>	46	<code>gsl_sf_coupling_3j</code>	55
<code>gsl_sf_bessel_Yn_e</code>	45	<code>gsl_sf_coupling_3j_e</code>	55
<code>gsl_sf_bessel_Ynu</code>	51	<code>gsl_sf_coupling_6j</code>	56
<code>gsl_sf_bessel_Ynu_e</code>	51	<code>gsl_sf_coupling_6j_e</code>	56
<code>gsl_sf_bessel_zero_J0</code>	52	<code>gsl_sf_coupling_9j</code>	56
<code>gsl_sf_bessel_zero_J0_e</code>	52	<code>gsl_sf_coupling_9j_e</code>	56
<code>gsl_sf_bessel_zero_J1</code>	52	<code>gsl_sf_dawson</code>	56
<code>gsl_sf_bessel_zero_J1_e</code>	52	<code>gsl_sf_dawson_e</code>	56
<code>gsl_sf_bessel_zero_Jnu</code>	52	<code>gsl_sf_debye_1</code>	57
<code>gsl_sf_bessel_zero_Jnu_e</code>	52	<code>gsl_sf_debye_1_e</code>	57
<code>gsl_sf_beta</code>	71	<code>gsl_sf_debye_2</code>	57
<code>gsl_sf_beta_e</code>	71	<code>gsl_sf_debye_2_e</code>	57
<code>gsl_sf_beta_inc</code>	72	<code>gsl_sf_debye_3</code>	57
<code>gsl_sf_beta_inc_e</code>	72	<code>gsl_sf_debye_3_e</code>	57

<code>gsl_sf_debye_4</code>	57	<code>gsl_sf_exp_err_e10_e</code>	64
<code>gsl_sf_debye_4_e</code>	57	<code>gsl_sf_exp_mult</code>	63
<code>gsl_sf_debye_5</code>	57	<code>gsl_sf_exp_mult_e</code>	63
<code>gsl_sf_debye_5_e</code>	57	<code>gsl_sf_exp_mult_e10_e</code>	63
<code>gsl_sf_debye_6</code>	57	<code>gsl_sf_exp_mult_err_e</code>	64
<code>gsl_sf_debye_6_e</code>	57	<code>gsl_sf_exp_mult_err_e10_e</code>	65
<code>gsl_sf_dilog</code>	58	<code>gsl_sf_expint_3</code>	66
<code>gsl_sf_dilog_e</code>	58	<code>gsl_sf_expint_3_e</code>	66
<code>gsl_sf_doublefact</code>	70	<code>gsl_sf_expint_E1</code>	65
<code>gsl_sf_doublefact_e</code>	70	<code>gsl_sf_expint_E1_e</code>	65
<code>gsl_sf_ellint_D</code>	61	<code>gsl_sf_expint_E2</code>	65
<code>gsl_sf_ellint_D_e</code>	61	<code>gsl_sf_expint_E2_e</code>	65
<code>gsl_sf_ellint_E</code>	60	<code>gsl_sf_expint_Ei</code>	65
<code>gsl_sf_ellint_E_e</code>	60	<code>gsl_sf_expint_Ei_e</code>	65
<code>gsl_sf_ellint_Ecomp</code>	60	<code>gsl_sf_expint_En</code>	65
<code>gsl_sf_ellint_Ecomp_e</code>	60	<code>gsl_sf_expint_En_e</code>	65
<code>gsl_sf_ellint_F</code>	60	<code>gsl_sf_expm1</code>	64
<code>gsl_sf_ellint_F_e</code>	60	<code>gsl_sf_expm1_e</code>	64
<code>gsl_sf_ellint_Kcomp</code>	60	<code>gsl_sf_exprel</code>	64
<code>gsl_sf_ellint_Kcomp_e</code>	60	<code>gsl_sf_exprel_2</code>	64
<code>gsl_sf_ellint_P</code>	60	<code>gsl_sf_exprel_2_e</code>	64
<code>gsl_sf_ellint_P_e</code>	60	<code>gsl_sf_exprel_e</code>	64
<code>gsl_sf_ellint_Pcomp</code>	60	<code>gsl_sf_exprel_n</code>	64
<code>gsl_sf_ellint_Pcomp_e</code>	60	<code>gsl_sf_exprel_n_e</code>	64
<code>gsl_sf_ellint_RC</code>	61	<code>gsl_sf_fact</code>	69
<code>gsl_sf_ellint_RC_e</code>	61	<code>gsl_sf_fact_e</code>	69
<code>gsl_sf_ellint_RD</code>	61	<code>gsl_sf_fermi_dirac_0</code>	67
<code>gsl_sf_ellint_RD_e</code>	61	<code>gsl_sf_fermi_dirac_0_e</code>	67
<code>gsl_sf_ellint_RF</code>	61	<code>gsl_sf_fermi_dirac_1</code>	67
<code>gsl_sf_ellint_RF_e</code>	61	<code>gsl_sf_fermi_dirac_1_e</code>	67
<code>gsl_sf_ellint_RJ</code>	61	<code>gsl_sf_fermi_dirac_2</code>	67
<code>gsl_sf_ellint_RJ_e</code>	61	<code>gsl_sf_fermi_dirac_2_e</code>	67
<code>gsl_sf_elljac_e</code>	62	<code>gsl_sf_fermi_dirac_3half</code>	68
<code>gsl_sf_erf</code>	62	<code>gsl_sf_fermi_dirac_3half_e</code>	68
<code>gsl_sf_erf_e</code>	62	<code>gsl_sf_fermi_dirac_half</code>	67
<code>gsl_sf_erf_Q</code>	62	<code>gsl_sf_fermi_dirac_half_e</code>	67
<code>gsl_sf_erf_Q_e</code>	62	<code>gsl_sf_fermi_dirac_inc_0</code>	68
<code>gsl_sf_erf_Z</code>	62	<code>gsl_sf_fermi_dirac_inc_0_e</code>	68
<code>gsl_sf_erf_Z_e</code>	62	<code>gsl_sf_fermi_dirac_int</code>	67
<code>gsl_sf_erfc</code>	62	<code>gsl_sf_fermi_dirac_int_e</code>	67
<code>gsl_sf_erfc_e</code>	62	<code>gsl_sf_fermi_dirac_m1</code>	67
<code>gsl_sf_eta</code>	86	<code>gsl_sf_fermi_dirac_m1_e</code>	67
<code>gsl_sf_eta_e</code>	86	<code>gsl_sf_fermi_dirac_mhalf</code>	67
<code>gsl_sf_eta_int</code>	86	<code>gsl_sf_fermi_dirac_mhalf_e</code>	67
<code>gsl_sf_eta_int_e</code>	86	<code>gsl_sf_gamma</code>	68
<code>gsl_sf_exp</code>	63	<code>gsl_sf_gamma_e</code>	68
<code>gsl_sf_exp_e</code>	63	<code>gsl_sf_gamma_inc</code>	71
<code>gsl_sf_exp_e10_e</code>	63	<code>gsl_sf_gamma_inc_e</code>	71
<code>gsl_sf_exp_err_e</code>	64	<code>gsl_sf_gamma_inc_P</code>	71

<code>gsl_sf_gamma_inc_P_e</code>	71	<code>gsl_sf_laguerre_1_e</code>	75
<code>gsl_sf_gamma_inc_Q</code>	71	<code>gsl_sf_laguerre_2</code>	75
<code>gsl_sf_gamma_inc_Q_e</code>	71	<code>gsl_sf_laguerre_2_e</code>	75
<code>gsl_sf_gammainv</code>	69	<code>gsl_sf_laguerre_3</code>	75
<code>gsl_sf_gammainv_e</code>	69	<code>gsl_sf_laguerre_3_e</code>	75
<code>gsl_sf_gammastar</code>	69	<code>gsl_sf_laguerre_n</code>	75
<code>gsl_sf_gammastar_e</code>	69	<code>gsl_sf_laguerre_n_e</code>	75
<code>gsl_sf_gegenpoly_1</code>	72	<code>gsl_sf_lambert_W0</code>	75
<code>gsl_sf_gegenpoly_1_e</code>	72	<code>gsl_sf_lambert_W0_e</code>	75
<code>gsl_sf_gegenpoly_2</code>	72	<code>gsl_sf_lambert_Wm1</code>	75
<code>gsl_sf_gegenpoly_2_e</code>	72	<code>gsl_sf_lambert_Wm1_e</code>	75
<code>gsl_sf_gegenpoly_3</code>	72	<code>gsl_sf_legendre_array_size</code>	77
<code>gsl_sf_gegenpoly_3_e</code>	72	<code>gsl_sf_legendre_H3d</code>	79
<code>gsl_sf_gegenpoly_array</code>	72	<code>gsl_sf_legendre_H3d_0</code>	78
<code>gsl_sf_gegenpoly_n</code>	72	<code>gsl_sf_legendre_H3d_0_e</code>	78
<code>gsl_sf_gegenpoly_n_e</code>	72	<code>gsl_sf_legendre_H3d_1</code>	79
<code>gsl_sf_hazard</code>	63	<code>gsl_sf_legendre_H3d_1_e</code>	79
<code>gsl_sf_hazard_e</code>	63	<code>gsl_sf_legendre_H3d_array</code>	79
<code>gsl_sf_hydrogenicR</code>	53	<code>gsl_sf_legendre_H3d_e</code>	79
<code>gsl_sf_hydrogenicR_1</code>	53	<code>gsl_sf_legendre_P1</code>	76
<code>gsl_sf_hydrogenicR_1_e</code>	53	<code>gsl_sf_legendre_P1_e</code>	76
<code>gsl_sf_hydrogenicR_e</code>	53	<code>gsl_sf_legendre_P2</code>	76
<code>gsl_sf_hyperg_0F1</code>	73	<code>gsl_sf_legendre_P2_e</code>	76
<code>gsl_sf_hyperg_0F1_e</code>	73	<code>gsl_sf_legendre_P3</code>	76
<code>gsl_sf_hyperg_1F1</code>	73	<code>gsl_sf_legendre_P3_e</code>	76
<code>gsl_sf_hyperg_1F1_e</code>	73	<code>gsl_sf_legendre_Pl</code>	76
<code>gsl_sf_hyperg_1F1_int</code>	73	<code>gsl_sf_legendre_Pl_array</code>	76
<code>gsl_sf_hyperg_1F1_int_e</code>	73	<code>gsl_sf_legendre_Pl_deriv_array</code> ..	76
<code>gsl_sf_hyperg_2F0</code>	74	<code>gsl_sf_legendre_Pl_e</code>	76
<code>gsl_sf_hyperg_2F0_e</code>	74	<code>gsl_sf_legendre_Plm</code>	77
<code>gsl_sf_hyperg_2F1</code>	73	<code>gsl_sf_legendre_Plm_array</code>	77
<code>gsl_sf_hyperg_2F1_conj</code>	74	<code>gsl_sf_legendre_Plm_deriv_array</code> ..	77
<code>gsl_sf_hyperg_2F1_conj_e</code>	74	<code>gsl_sf_legendre_Plm_e</code>	77
<code>gsl_sf_hyperg_2F1_conj_renorm</code> ..	74	<code>gsl_sf_legendre_Q0</code>	76
<code>gsl_sf_hyperg_2F1_conj_renorm_e</code> ..	74	<code>gsl_sf_legendre_Q0_e</code>	76
<code>gsl_sf_hyperg_2F1_e</code>	73	<code>gsl_sf_legendre_Q1</code>	76
<code>gsl_sf_hyperg_2F1_renorm</code>	74	<code>gsl_sf_legendre_Q1_e</code>	76
<code>gsl_sf_hyperg_2F1_renorm_e</code>	74	<code>gsl_sf_legendre_Ql</code>	76
<code>gsl_sf_hyperg_U</code>	73	<code>gsl_sf_legendre_Ql_e</code>	76
<code>gsl_sf_hyperg_U_e</code>	73	<code>gsl_sf_legendre_sphPlm</code>	77
<code>gsl_sf_hyperg_U_e10_e</code>	73	<code>gsl_sf_legendre_sphPlm_array</code>	77
<code>gsl_sf_hyperg_U_int</code>	73	<code>gsl_sf_legendre_sphPlm_deriv_array</code>	
<code>gsl_sf_hyperg_U_int_e</code>	73	77
<code>gsl_sf_hyperg_U_int_e10_e</code>	73	<code>gsl_sf_legendre_sphPlm_e</code>	77
<code>gsl_sf_hypot</code>	84	<code>gsl_sf_lnbeta</code>	71
<code>gsl_sf_hypot_e</code>	84	<code>gsl_sf_lnbeta_e</code>	71
<code>gsl_sf_hzeta</code>	86	<code>gsl_sf_lnchoose</code>	70
<code>gsl_sf_hzeta_e</code>	86	<code>gsl_sf_lnchoose_e</code>	70
<code>gsl_sf_laguerre_1</code>	75	<code>gsl_sf_lncosh</code>	85

<code>gsl_sf_lncosh_e</code>	85	<code>gsl_sf_psi_1_e</code>	83
<code>gsl_sf_lndoublefact</code>	70	<code>gsl_sf_psi_1_int</code>	83
<code>gsl_sf_lndoublefact_e</code>	70	<code>gsl_sf_psi_1_int_e</code>	83
<code>gsl_sf_lnfact</code>	70	<code>gsl_sf_psi_1piy</code>	83
<code>gsl_sf_lnfact_e</code>	70	<code>gsl_sf_psi_1piy_e</code>	83
<code>gsl_sf_lngamma</code>	68	<code>gsl_sf_psi_e</code>	82
<code>gsl_sf_lngamma_complex_e</code>	69	<code>gsl_sf_psi_int</code>	82
<code>gsl_sf_lngamma_e</code>	68	<code>gsl_sf_psi_int_e</code>	82
<code>gsl_sf_lngamma_sgn_e</code>	69	<code>gsl_sf_psi_n</code>	83
<code>gsl_sf_lnpoch</code>	70	<code>gsl_sf_psi_n_e</code>	83
<code>gsl_sf_lnpoch_e</code>	70	<code>gsl_sf_rect_to_polar</code>	85
<code>gsl_sf_lnpoch_sgn_e</code>	71	<code>gsl_sf_Shi</code>	66
<code>gsl_sf_lnsinh</code>	85	<code>gsl_sf_Shi_e</code>	66
<code>gsl_sf_lnsinh_e</code>	85	<code>gsl_sf_Si</code>	66
<code>gsl_sf_log</code>	79	<code>gsl_sf_Si_e</code>	66
<code>gsl_sf_log_1plusx</code>	79	<code>gsl_sf_sin</code>	84
<code>gsl_sf_log_1plusx_e</code>	79	<code>gsl_sf_sin_e</code>	84
<code>gsl_sf_log_1plusx_mx</code>	80	<code>gsl_sf_sin_err_e</code>	85
<code>gsl_sf_log_1plusx_mx_e</code>	80	<code>gsl_sf_sinc</code>	84
<code>gsl_sf_log_abs</code>	79	<code>gsl_sf_sinc_e</code>	84
<code>gsl_sf_log_abs_e</code>	79	<code>gsl_sf_synchrotron_1</code>	83
<code>gsl_sf_log_e</code>	79	<code>gsl_sf_synchrotron_1_e</code>	83
<code>gsl_sf_log_erfc</code>	62	<code>gsl_sf_synchrotron_2</code>	83
<code>gsl_sf_log_erfc_e</code>	62	<code>gsl_sf_synchrotron_2_e</code>	83
<code>gsl_sf_mathieu_a</code>	81	<code>gsl_sf_taylorcoeff</code>	70
<code>gsl_sf_mathieu_a_array</code>	81	<code>gsl_sf_taylorcoeff_e</code>	70
<code>gsl_sf_mathieu_alloc</code>	80	<code>gsl_sf_transport_2</code>	83
<code>gsl_sf_mathieu_b</code>	81	<code>gsl_sf_transport_2_e</code>	83
<code>gsl_sf_mathieu_b_array</code>	81	<code>gsl_sf_transport_3</code>	83
<code>gsl_sf_mathieu_ce</code>	81	<code>gsl_sf_transport_3_e</code>	83
<code>gsl_sf_mathieu_ce_array</code>	81	<code>gsl_sf_transport_4</code>	84
<code>gsl_sf_mathieu_free</code>	80	<code>gsl_sf_transport_4_e</code>	84
<code>gsl_sf_mathieu_Mc</code>	81	<code>gsl_sf_transport_5</code>	84
<code>gsl_sf_mathieu_Mc_array</code>	82	<code>gsl_sf_transport_5_e</code>	84
<code>gsl_sf_mathieu_Ms</code>	81	<code>gsl_sf_zeta</code>	86
<code>gsl_sf_mathieu_Ms_array</code>	82	<code>gsl_sf_zeta_e</code>	86
<code>gsl_sf_mathieu_se</code>	81	<code>gsl_sf_zeta_int</code>	86
<code>gsl_sf_mathieu_se_array</code>	81	<code>gsl_sf_zeta_int_e</code>	86
<code>gsl_sf_multiply_e</code>	58	<code>gsl_sf_zetam1</code>	86
<code>gsl_sf_multiply_err_e</code>	58	<code>gsl_sf_zetam1_e</code>	86
<code>gsl_sf_poch</code>	70	<code>gsl_sf_zetam1_int</code>	86
<code>gsl_sf_poch_e</code>	70	<code>gsl_sf_zetam1_int_e</code>	86
<code>gsl_sf_pochrel</code>	71	<code>GSL_SIGN</code>	24
<code>gsl_sf_pochrel_e</code>	71	<code>gsl_siman_solve</code>	338
<code>gsl_sf_polar_to_rect</code>	85	<code>gsl_sort</code>	132
<code>gsl_sf_pow_int</code>	82	<code>gsl_sort_index</code>	132
<code>gsl_sf_pow_int_e</code>	82	<code>gsl_sort_largest</code>	133
<code>gsl_sf_psi</code>	82	<code>gsl_sort_largest_index</code>	134
<code>gsl_sf_psi_1</code>	83	<code>gsl_sort_smallest</code>	133

<code>gsl_sort_smallest_index</code>	133	<code>gsl_stats_variance_m</code>	288
<code>gsl_sort_vector</code>	132	<code>gsl_stats_variance_with_fixed_mean</code>	288
<code>gsl_sort_vector_index</code>	133	<code>gsl_stats_wabsdev</code>	293
<code>gsl_sort_vector_largest</code>	133	<code>gsl_stats_wabsdev_m</code>	293
<code>gsl_sort_vector_largest_index</code> ...	134	<code>gsl_stats_wkurtosis</code>	293
<code>gsl_sort_vector_smallest</code>	133	<code>gsl_stats_wkurtosis_m_sd</code>	294
<code>gsl_sort_vector_smallest_index</code> ..	134	<code>gsl_stats_wmean</code>	291
<code>gsl_spline_alloc</code>	362	<code>gsl_stats_wsd</code>	292
<code>gsl_spline_eval</code>	362	<code>gsl_stats_wsd_m</code>	292
<code>gsl_spline_eval_deriv</code>	362	<code>gsl_stats_wsd_with_fixed_mean</code> ...	292
<code>gsl_spline_eval_deriv_e</code>	362	<code>gsl_stats_wskew</code>	293
<code>gsl_spline_eval_deriv2</code>	363	<code>gsl_stats_wskew_m_sd</code>	293
<code>gsl_spline_eval_deriv2_e</code>	363	<code>gsl_stats_wtss</code>	293
<code>gsl_spline_eval_e</code>	362	<code>gsl_stats_wtss_m</code>	293
<code>gsl_spline_eval_integ</code>	363	<code>gsl_stats_wvariance</code>	292
<code>gsl_spline_eval_integ_e</code>	363	<code>gsl_stats_wvariance_m</code>	292
<code>gsl_spline_free</code>	362	<code>gsl_stats_wvariance_with_fixed_mean</code>	292
<code>gsl_spline_init</code>	362	<code>gsl_strerror</code>	16
<code>gsl_spline_min_size</code>	362	<code>gsl_sum_levin_u_accel</code>	375
<code>gsl_spline_name</code>	362	<code>gsl_sum_levin_u_alloc</code>	375
<code>gsl_stats_absdev</code>	289	<code>gsl_sum_levin_u_free</code>	375
<code>gsl_stats_absdev_m</code>	289	<code>gsl_sum_levin_utrunc_accel</code>	376
<code>gsl_stats_correlation</code>	291	<code>gsl_sum_levin_utrunc_alloc</code>	376
<code>gsl_stats_covariance</code>	291	<code>gsl_sum_levin_utrunc_free</code>	376
<code>gsl_stats_covariance_m</code>	291	<code>gsl_vector_add</code>	98
<code>gsl_stats_kurtosis</code>	290	<code>gsl_vector_add_constant</code>	98
<code>gsl_stats_kurtosis_m_sd</code>	290	<code>gsl_vector_alloc</code>	92
<code>gsl_stats_lag1_autocorrelation</code> ..	290	<code>gsl_vector_calloc</code>	92
<code>gsl_stats_lag1_autocorrelation_m</code>	290	<code>gsl_vector_complex_const_imag</code>	96
<code>gsl_stats_max</code>	294	<code>gsl_vector_complex_const_real</code>	96
<code>gsl_stats_max_index</code>	294	<code>gsl_vector_complex_imag</code>	96
<code>gsl_stats_mean</code>	287	<code>gsl_vector_complex_real</code>	96
<code>gsl_stats_median_from_sorted_data</code>	295	<code>gsl_vector_const_ptr</code>	93
<code>gsl_stats_min</code>	294	<code>gsl_vector_const_subvector</code>	95
<code>gsl_stats_min_index</code>	294	<code>gsl_vector_const_subvector_with_ stride</code>	95
<code>gsl_stats_minmax</code>	294	<code>gsl_vector_const_view_array</code>	96
<code>gsl_stats_minmax_index</code>	295	<code>gsl_vector_const_view_array_with_ stride</code>	97
<code>gsl_stats_quantile_from_sorted_data</code>	295	<code>gsl_vector_div</code>	98
<code>gsl_stats_sd</code>	288	<code>gsl_vector_fprintf</code>	94
<code>gsl_stats_sd_m</code>	288	<code>gsl_vector_fread</code>	94
<code>gsl_stats_sd_with_fixed_mean</code>	288	<code>gsl_vector_free</code>	92
<code>gsl_stats_skew</code>	289	<code>gsl_vector_fscanf</code>	94
<code>gsl_stats_skew_m_sd</code>	289	<code>gsl_vector_fwrite</code>	94
<code>gsl_stats_tss</code>	288	<code>gsl_vector_get</code>	93
<code>gsl_stats_tss_m</code>	288	<code>gsl_vector_isneg</code>	99
<code>gsl_stats_variance</code>	287		

<code>gsl_vector_isnonneg</code>	99	<code>gsl_wavelet_daubechies_centered</code>	
<code>gsl_vector_isnull</code>	99	379
<code>gsl_vector_ispos</code>	99	<code>gsl_wavelet_free</code>	380
<code>gsl_vector_max</code>	98	<code>gsl_wavelet_haar</code>	380
<code>gsl_vector_max_index</code>	99	<code>gsl_wavelet_haar_centered</code>	380
<code>gsl_vector_memcpy</code>	97	<code>gsl_wavelet_name</code>	380
<code>gsl_vector_min</code>	98	<code>gsl_wavelet_transform</code>	381
<code>gsl_vector_min_index</code>	99	<code>gsl_wavelet_transform_forward</code> ...	381
<code>gsl_vector_minmax</code>	98	<code>gsl_wavelet_transform_inverse</code> ...	381
<code>gsl_vector_minmax_index</code>	99	<code>gsl_wavelet_workspace_alloc</code>	380
<code>gsl_vector_mul</code>	98	<code>gsl_wavelet_workspace_free</code>	380
<code>gsl_vector_ptr</code>	93	<code>gsl_wavelet2d_nstransform</code>	382
<code>gsl_vector_reverse</code>	98	<code>gsl_wavelet2d_nstransform_forward</code>	
<code>gsl_vector_scale</code>	98	382
<code>gsl_vector_set</code>	93	<code>gsl_wavelet2d_nstransform_inverse</code>	
<code>gsl_vector_set_all</code>	94	382
<code>gsl_vector_set_basis</code>	94	<code>gsl_wavelet2d_nstransform_matrix</code>	
<code>gsl_vector_set_zero</code>	94	383
<code>gsl_vector_sub</code>	98	<code>gsl_wavelet2d_nstransform_matrix_</code>	
<code>gsl_vector_subvector</code>	95	<code>forward</code>	383
<code>gsl_vector_subvector_with_stride</code>		<code>gsl_wavelet2d_nstransform_matrix_</code>	
.....	95	<code>inverse</code>	383
<code>gsl_vector_swap</code>	97	<code>gsl_wavelet2d_transform</code>	382
<code>gsl_vector_swap_elements</code>	98	<code>gsl_wavelet2d_transform_forward</code>	
<code>gsl_vector_view_array</code>	96	382
<code>gsl_vector_view_array_with_stride</code>		<code>gsl_wavelet2d_transform_inverse</code>	
.....	97	382
<code>gsl_wavelet_alloc</code>	379	<code>gsl_wavelet2d_transform_matrix</code> ..	382
<code>gsl_wavelet_bspline</code>	380	<code>gsl_wavelet2d_transform_matrix_</code>	
<code>gsl_wavelet_bspline_centered</code>	380	<code>forward</code>	382
<code>gsl_wavelet_daubechies</code>	379	<code>gsl_wavelet2d_transform_matrix_</code>	
		<code>inverse</code>	382

Type and Variable Index

A

alpha 329, 332

C

chisq 332

D

dither 330

E

estimate_frac 329

G

gsl_block 89
 gsl_bspline_deriv_workspace 470
 gsl_bspline_workspace 470
 GSL_C99_INLINE 9, 93
 gsl_cheb_series 371
 gsl_check_range 93
 gsl_combination 125
 gsl_complex 27
 gsl_dht 388
 GSL_EDOM 16
 gsl_eigen_gen_workspace 178
 gsl_eigen_genherm_workspace 176
 gsl_eigen_genhermv_workspace 176
 gsl_eigen_gensymm_workspace 175
 gsl_eigen_gensymmv_workspace 175
 gsl_eigen_genv_workspace 178
 gsl_eigen_herm_workspace 172
 gsl_eigen_hermv_workspace 172
 gsl_eigen_nonsymm_workspace 173
 gsl_eigen_nonsymmv_workspace 174
 gsl_eigen_symm_workspace 171
 gsl_eigen_symmv_workspace 171
 GSL_EINVAL 16
 GSL_ENOMEM 16
 GSL_ERANGE 16
 gsl_error_handler_t 17
 gsl_fft_complex_wavetable 191

gsl_fft_complex_workspace 191
 gsl_fft_halfcomplex_wavetable 198
 gsl_fft_real_wavetable 198
 gsl_fft_real_workspace 198
 gsl_function 391
 gsl_function_fdf 392
 gsl_histogram 299
 gsl_histogram_pdf 306
 gsl_histogram2d 309
 gsl_histogram2d_pdf 316
 GSL_IEEE_MODE 489
 gsl_integration_qawo_table 212
 gsl_integration_qaws_table 211
 gsl_integration_workspace 207
 gsl_interp 359
 gsl_interp_accel 361
 gsl_interp_type 360
 gsl_matrix 101
 gsl_matrix_const_view 104
 gsl_matrix_view 104
 gsl_min_fminimizer 406
 gsl_min_fminimizer_type 406
 gsl_monte_function 325
 gsl_monte_miser_state 328
 gsl_monte_plain_state 327
 gsl_monte_vegas_state 331
 gsl_multifit_fdfsolver 456
 gsl_multifit_fdfsolver_type 456
 gsl_multifit_fsolver 456
 gsl_multifit_fsolver_type 456
 gsl_multifit_function 457
 gsl_multifit_function_fdf 457
 gsl_multifit_linear_workspace 446
 gsl_multimin_fdfminimizer 430
 gsl_multimin_fdfminimizer_type 430
 gsl_multimin_fminimizer 430
 gsl_multimin_fminimizer_type 430
 gsl_multimin_function 432
 gsl_multimin_function_fdf 431
 gsl_multisearch_fdfsolver 414
 gsl_multisearch_fdfsolver_type 414
 gsl_multisearch_fsolver 414
 gsl_multisearch_fsolver_type 414
 gsl_multisearch_function 415
 gsl_multisearch_function_fdf 416

Concept Index

\$

\$, shell prompt 6

2

2D histograms 308

2D random direction vector 266

3

3-j symbols 55

3D random direction vector 266

6

6-j symbols 55

9

9-j symbols 55

A

acceleration of series 375

acosh 23

Adaptive step-size control, differential
equations 350

$Ai(x)$ 43

Airy functions 43

Akima splines 360

aliasing of arrays 13

alternative optimized functions 11

AMAX, Level-1 BLAS 140

Angular Mathieu Functions 81

angular reduction 85

ANSI C, use of 7

Apell symbol, see Pochhammer symbol
..... 70

approximate comparison of floating
point numbers 26

arctangent integral 66

argument of complex number 28

arithmetic exceptions 489

asinh 23

astronomical constants 478

ASUM, Level-1 BLAS 139

atanh 23

atomic physics, constants 478

autoconf, using with GSL 501

AXPY, Level-1 BLAS 140

B

B-spline wavelets 380

Bader and Deuffhard, Bulirsch-Stoer
method 349

balancing matrices 167

Basic Linear Algebra Subroutines
(BLAS) 137, 503

basis splines, B-splines 469

basis splines, derivatives 471

basis splines, evaluation 471

basis splines, examples 472

basis splines, initializing 470

basis splines, overview 469

Bernoulli trial, random variates 274

Bessel functions 45

Bessel Functions, Fractional Order ... 51

best-fit parameters, covariance 461

Beta distribution 263

Beta function 71

Beta function, incomplete normalized
..... 72

BFGS algorithm, minimization 435

$Bi(x)$ 43

bias, IEEE format 487

biagonalization of real matrices ... 164

binning data 299

Binomial random variates 275

biorthogonal wavelets 380

bisection algorithm for finding roots
..... 396

Bivariate Gaussian distribution 245

BLAS 137

BLAS, Low-level C interface 503

blocks 89

bounds checking, extension to GCC .. 93

breakpoints 493

Brent's method for finding minima .. 409

Brent's method for finding roots 396
 Broyden algorithm for multidimensional
 roots 422
 BSD random number generator 227
 bug-gsl mailing list 5
 bugs, how to report 5
 Bulirsch-Stoer method 349

C

C extensions, compatible use of 7
 C++, compatibility 13
 C99, inline keyword 9
 Carlson forms of Elliptic integrals . . . 59
 Cash-Karp, Runge-Kutta method . . . 349
 Cauchy distribution 249
 Cauchy principal value, by numerical
 quadrature 210
 CBLAS 137
 CBLAS, Low-level interface 503
 CDFs, cumulative distribution functions
 239
ce(q, x), Mathieu function 81
 Chebyshev series 371
 checking combination for validity . . . 126
 checking permutation for validity . . . 118
 Chi(x) 66
 Chi-squared distribution 259
 Cholesky decomposition 160
 Ci(x) 66
 Clausen functions 53
 Clenshaw-Curtis quadrature 206
 CMRG, combined multiple recursive
 random number generator 225
 code reuse in applications 14
 combinations 125
 combinatorial factor $C(m, n)$ 70
 combinatorial optimization 337
 comparison functions, definition . . . 131
 compatibility 7
 compiling programs, include paths . . . 7
 compiling programs, library paths . . . 8
 complementary incomplete Gamma
 function 71
 complete Fermi-Dirac integrals 67
 complex arithmetic 28
 complex cosine function, special
 functions 84
 Complex Gamma function 69

complex hermitian matrix, eigensystem
 172
 complex log sine function, special
 functions 84
 complex numbers 27
 complex sinc function, special functions
 84
 complex sine function, special functions
 84
 confluent hypergeometric function . . . 74
 confluent hypergeometric functions . . 73
 conical functions 75
 Conjugate gradient algorithm,
 minimization 435
 conjugate of complex number 29
 constant matrix 103
 constants, fundamental 477
 constants, mathematical—defined as
 macros 21
 constants, physical 477
 constants, prefixes 483
 contacting the GSL developers 6
 conventions, used in manual 6
 convergence, accelerating a series . . . 375
 conversion of units 477
 cooling schedule 337
 COPY, Level-1 BLAS 140
 correlation, of two datasets 291
 cosine function, special functions 84
 cosine of complex number 30
 cost function 337
 Coulomb wave functions 53
 coupling coefficients 55
 covariance matrix, from linear regression
 444
 covariance matrix, linear fits 443
 covariance matrix, nonlinear fits . . . 461
 covariance, of two datasets 291
 CRAY random number generator,
 RANF 228
 cubic equation, solving 37
 cubic splines 360
 cumulative distribution functions
 (CDFs) 239
 Cylindrical Bessel Functions 45

D

Daubechies wavelets 379

Dawson function	56
DAXPY, Level-1 BLAS	140
debugging numerical programs.....	493
Debye functions	57
denormalized form, IEEE format ...	487
deprecated functions.....	14
derivatives, calculating numerically	367
determinant of a matrix, by LU decomposition	154
Deuffhard and Bader, Bulirsch-Stoer method.	349
DFTs, see FFT	185
diagonal, of a matrix	108
differential equations, initial value problems	347
differentiation of functions, numeric	367
digamma function	82
dilogarithm.....	58
direction vector, random 2D	266
direction vector, random 3D	266
direction vector, random N-dimensional	266
Dirichlet distribution	270
discontinuities, in ODE systems ...	352
Discrete Fourier Transforms, see FFT	185
discrete Hankel transforms.....	387
Discrete Newton algorithm for multidimensional roots.....	422
Discrete random numbers.....	271, 272
Discrete random numbers, preprocessing	271
divided differences, polynomials.....	35
division by zero, IEEE exceptions...	489
dollar sign \$, shell prompt	6
DOT, Level-1 BLAS	139
double factorial	70
double precision, IEEE format.....	487
downloading GSL.....	4
DWT initialization	379
DWT, mathematical definition	379
DWT, one dimensional	381
DWT, see wavelet transforms.....	379
DWT, two dimensional.....	381

E

e, defined as a macro	21
E1(x), E2(x), Ei(x)	65
eigenvalues and eigenvectors	171
elementary functions	21
elementary operations	58
elliptic functions (Jacobi)	62
elliptic integrals	59
energy function.....	337
energy, units of.....	481
erf(x)	62
erfc(x)	62
Erlang distribution	255
error codes	16
error codes, reserved.....	16
error function	62
Error handlers.....	17
error handling	15
error handling macros	18
Errors.....	15
estimated standard deviation	287
estimated variance	287
Eta Function	86
euclidean distance function, hypot ...	23
Euler's constant, defined as a macro..	22
evaluation of polynomials.....	35
evaluation of polynomials, in divided difference form.....	35
examples, conventions used in.....	6
exceptions, C++	13
exceptions, floating point	495
exceptions, IEEE arithmetic.....	489
exchanging permutation elements...	118
exp	63
expm1	23
exponent, IEEE format	487
Exponential distribution.....	246
exponential function.....	63
exponential integrals.....	65
Exponential power distribution	248
exponential, difference from 1 computed accurately	23
exponentiation of complex number...	30
extern inline	9

F

F-distribution	260
factorial	69

factorization of matrices 153

false position algorithm for finding roots
..... 396

Fast Fourier Transforms, see FFT .. 185

Fehlberg method, differential equations
..... 349

Fermi-Dirac function 67

FFT 185

FFT mathematical definition 185

FFT of complex data, mixed-radix
algorithm 190

FFT of complex data, radix-2 algorithm
..... 187

FFT of real data 194

FFT of real data, mixed-radix algorithm
..... 197

FFT of real data, radix-2 algorithm
..... 195

FFT, complex data 186

finding minima 405

finding roots 389

finding zeros 389

fits, multi-parameter linear 446

fitting 443

fitting, using Chebyshev polynomials
..... 371

Fj(x), Fermi-Dirac integral 67

Fj(x,b), incomplete Fermi-Dirac integral
..... 68

flat distribution 257

Fletcher-Reeves conjugate gradient
algorithm, minimization 435

floating point exceptions 495

floating point numbers, approximate
comparison 26

floating point registers 494

force and energy, units of 483

Fortran range checking, equivalent in gcc
..... 93

Four-tap Generalized Feedback Shift
Register 226

Fourier integrals, numerical 213

Fourier Transforms, see FFT 185

Fractional Order Bessel Functions ... 51

free documentation 519

free software, explanation of 4

frexp 23

functions, numerical differentiation
..... 367

fundamental constants 477

G

Gamma distribution 255

gamma functions 68

Gauss-Kronrod quadrature 206

Gaussian distribution 241

Gaussian distribution, bivariate 245

Gaussian Tail distribution 243

gcc extensions, range-checking 93

gcc warning options 495

gdb 493

Gear method, differential equations
..... 349

Gegenbauer functions 72

GEMM, Level-3 BLAS 145

GEMV, Level-2 BLAS 142

general polynomial equations, solving
..... 37

generalized eigensystems 177

generalized hermitian definite
eigensystems 176

generalized symmetric eigensystems
..... 175

Geometric random variates 279, 280

GER, Level-2 BLAS 144

GERC, Level-2 BLAS 144

GERU, Level-2 BLAS 144

Givens Rotation, BLAS 141

Givens Rotation, Modified, BLAS .. 141

GNU General Public License 3

golden section algorithm for finding
minima 409

GSL_C99_INLINE 9

GSL_RNG_SEED 218

gsl_sf_result 42

gsl_sf_result_e10 42

Gumbel distribution (Type 1) 268

Gumbel distribution (Type 2) 269

H

Haar wavelets 380

Hankel transforms, discrete 387

HAVE_INLINE 9

hazard function, normal distribution
..... 63

HBOOK 324

header files, including.....	7	IEEE floating point	487
heapsort	131	IEEE format for floating point numbers	487
HEMM, Level-3 BLAS	146	IEEE infinity, defined as a macro	22
HEMV, Level-2 BLAS.....	143	IEEE NaN, defined as a macro.....	22
HER, Level-2 BLAS	144	illumination, units of.....	482
HER2, Level-2 BLAS.....	145	imperial units.....	479
HER2K, Level-3 BLAS.....	149	importance sampling, VEGAS.....	330
HERK, Level-3 BLAS	148	including GSL header files.....	7
hermitian matrix, complex, eigensystem	172	incomplete Beta function, normalized	72
hessenberg decomposition	163	incomplete Fermi-Dirac integral.....	68
hessenberg triangular decomposition	164	incomplete Gamma function.....	71
histogram statistics.....	303	indirect sorting.....	132
histogram, from ntuple	320	indirect sorting, of vector elements..	132
histograms	299	infinity, defined as a macro	22
histograms, random sampling from..	306	infinity, IEEE format	487
Householder linear solver	166	info-gsl mailing list	4
Householder matrix	165	initial value problems, differential	347
Householder transformation	165	equations	347
Hurwitz Zeta Function	86	initializing matrices	103
HYBRID algorithm, unscaled without	421	initializing vectors	94
HYBRID algorithms for nonlinear	420	inline functions.....	9
HYBRIDJ algorithm	421	integer powers.....	82
HYBRIDS algorithm, scaled without	421	integrals, exponential	65
HYBRIDSJ algorithm	420	integration, numerical (quadrature)	205
hydrogen atom	53	interpolation	359
hyperbolic cosine, inverse	23	interpolation, using Chebyshev	371
hyperbolic functions, complex numbers	32	polynomials.....	371
hyperbolic integrals.....	66	inverse complex trigonometric functions	31
hyperbolic sine, inverse	23	inverse cumulative distribution functions	239
hyperbolic space.....	75	inverse hyperbolic cosine.....	23
hyperbolic tangent, inverse	23	inverse hyperbolic functions, complex	33
hypergeometric functions	73	numbers	33
hypergeometric random variates	280	inverse hyperbolic sine.....	23
hypot	23	inverse hyperbolic tangent	23
hypot function, special functions.....	84	inverse of a matrix, by LU	154
		decomposition	154
		inverting a permutation	118
		Irregular Cylindrical Bessel Functions	45
		Irregular Modified Bessel Functions,	52
		Fractional Order	52
		Irregular Modified Cylindrical Bessel	47
		Functions	47

I

i(x), Bessel Functions.....	49
I(x), Bessel Functions.....	46
identity matrix.....	103
identity permutation	117
IEEE exceptions	489

Irregular Modified Spherical Bessel Functions	50
Irregular Spherical Bessel Functions ..	49
iterating through combinations	126
iterating through permutations	118
iterative refinement of solutions in linear systems	154

J

$j(x)$, Bessel Functions	48
$J(x)$, Bessel Functions	45
Jacobi elliptic functions	62
Jacobi orthogonalization	160
Jacobian matrix, fitting	455
Jacobian matrix, ODEs	347
Jacobian matrix, root finding	413

K

$k(x)$, Bessel Functions	50
$K(x)$, Bessel Functions	47
knots, basis splines	470
kurtosis	289

L

Laguerre functions	74
Lambert function	75
Landau distribution	252
LAPACK	184
Laplace distribution	247
LD_LIBRARY_PATH	8
ldexp	23
leading dimension, matrices	101
least squares fit	443
least squares fitting, nonlinear	455
least squares, covariance of best-fit parameters	461
Legendre forms of elliptic integrals ..	59
Legendre functions	75
Legendre polynomials	75
length, computed accurately using hypot	23
Levenberg-Marquardt algorithms ..	460
Levin u-transform	375
Levy distribution	253
Levy distribution, skew	254
libraries, linking with	8

libraries, shared	8
license of GSL	3
light, units of	482
linear algebra	153
linear algebra, BLAS	137
linear interpolation	360
linear regression	444
linear systems, refinement of solutions	154
linear systems, solution of	153
linking with GSL libraries	8
LMDER algorithm	460
log1p	23
logarithm and related functions	79
logarithm of Beta function	71
logarithm of combinatorial factor $C(m,n)$	70
logarithm of complex number	30
logarithm of cosh function, special functions	85
logarithm of double factorial	70
logarithm of factorial	70
logarithm of Gamma function	68
logarithm of Pochhammer symbol	70
logarithm of sinh function, special functions	85
logarithm of the determinant of a matrix	154
logarithm, computed accurately near 1	23
Logarithmic random variates	281
Logistic distribution	264
Lognormal distribution	258
long double	10
low discrepancy sequences	235
Low-level CBLAS	503
LU decomposition	153

M

macros for mathematical constants ..	21
magnitude of complex number	28
mailing list archives	6
mailing list for GSL announcements ..	4
mailing list, bug-gsl	5
mantissa, IEEE format	487
mass, units of	481
mathematical constants, defined as macros	21

mathematical functions, elementary.. 21
 Mathieu Function Characteristic Values
 81
 Mathieu functions..... 80
 matrices..... 89, 101
 matrices, initializing..... 103
 matrices, range-checking..... 103
 matrix determinant..... 154
 matrix diagonal..... 108
 matrix factorization..... 153
 matrix inverse..... 154
 matrix square root, Cholesky
 decomposition..... 160
 matrix subdiagonal..... 108
 matrix superdiagonal..... 108
 matrix, constant..... 103
 matrix, identity..... 103
 matrix, operations..... 137
 matrix, zero..... 103
 max..... 287
 maximal phase, Daubechies wavelets
 379
 maximization, see minimization.... 405
 maximum of two numbers..... 25
 maximum value, from histogram.... 303
 mean..... 287
 mean value, from histogram..... 303
 Mill's ratio, inverse..... 63
 min..... 287
 minimization, BFGS algorithm..... 435
 minimization, caveats..... 406
 minimization, conjugate gradient
 algorithm..... 435
 minimization, multidimensional.... 429
 minimization, one-dimensional..... 405
 minimization, overview..... 405
 minimization, Polak-Ribiere algorithm
 435
 minimization, providing a function to
 minimize..... 407
 minimization, simplex algorithm.... 436
 minimization, steepest descent algorithm
 435
 minimization, stopping parameters.. 408
 minimum finding, Brent's method.. 409
 minimum finding, golden section
 algorithm..... 409
 minimum of two numbers..... 25
 minimum value, from histogram.... 303

MINPACK, minimization algorithms
 420, 460
 MISCFUN..... 88
 MISER monte carlo integration.... 328
 Mixed-radix FFT, complex data.... 190
 Mixed-radix FFT, real data..... 197
 Modified Bessel Functions, Fractional
 Order..... 51
 Modified Clenshaw-Curtis quadrature
 206
 Modified Cylindrical Bessel Functions
 46
 Modified Givens Rotation, BLAS... 141
 Modified Newton's method for nonlinear
 systems..... 421
 Modified Spherical Bessel Functions.. 49
 Monte Carlo integration..... 325
 MRG, multiple recursive random
 number generator..... 225
 MT19937 random number generator
 223
 multi-parameter regression..... 446
 multidimensional integration..... 325
 multidimensional root finding, Broyden
 algorithm..... 422
 multidimensional root finding, overview
 413
 multidimensional root finding, providing
 a function to solve..... 415
 Multimin, caveats..... 430
 Multinomial distribution..... 276
 multiplication..... 58

N

N-dimensional random direction vector
 266
 NaN, defined as a macro..... 22
 nautical units..... 480
 Negative Binomial distribution, random
 variates..... 277
 Nelder-Mead simplex algorithm for
 minimization..... 436
 Newton algorithm, discrete..... 422
 Newton algorithm, globally convergent
 421
 Newton's method for finding roots.. 397
 Newton's method for systems of
 nonlinear equations..... 421

Niederreiter sequence.....	235
NIST Statistical Reference Datasets	453
non-normalized incomplete Gamma function	71
nonlinear equation, solutions of.....	389
nonlinear fitting, stopping parameters	459
nonlinear functions, minimization...	405
nonlinear least squares fitting.....	455
nonlinear least squares fitting, overview	455
nonlinear systems of equations, solution of	413
nonsymmetric matrix, real, eigensystem	173
normalized form, IEEE format.....	487
normalized incomplete Beta function	72
Not-a-number, defined as a macro ...	22
NRM2, Level-1 BLAS	139
ntuples.....	319
nuclear physics, constants	478
numerical constants, defined as macros	21
numerical derivatives	367
numerical integration (quadrature)	205

O

obtaining GSL.....	4
ODEs, initial value problems	347
optimization, combinatorial.....	337
optimization, see minimization.....	405
optimized functions, alternatives....	11
ordering, matrix elements.....	101
ordinary differential equations, initial value problem	347
oscillatory functions, numerical integration of.....	212
overflow, IEEE exceptions	489

P

Pareto distribution	265
PAW.....	324
permutations.....	117
physical constants.....	477

physical dimension, matrices.....	101
pi, defined as a macro	21
plain monte carlo.....	327
Pochhammer symbol	70
Poisson random numbers	273
Polak-Ribiere algorithm, minimization	435
polar form of complex numbers	27
polar to rectangular conversion	85
polygamma functions	82
polynomial evaluation	35
polynomial interpolation.....	360
polynomials, roots of	35
power function	82
power of complex number.....	30
power, units of	481
precision, IEEE arithmetic.....	489
prefixes	483
pressure, units of.....	482
Prince-Dormand, Runge-Kutta method	349
printers units	480
probability distribution, from histogram	306
probability distributions, from histograms.....	305
projection of ntuples	320
psi function	82

Q

QAG quadrature algorithm.....	207
QAGI quadrature algorithm	209
QAGP quadrature algorithm	209
QAGS quadrature algorithm.....	208
QAWC quadrature algorithm	210
QAWF quadrature algorithm	213
QAWO quadrature algorithm	212
QAWS quadrature algorithm	211
QNG quadrature algorithm.....	207
QR decomposition	155
QR decomposition with column pivoting	157
QUADPACK.....	205
quadratic equation, solving	36
quadrature	205
quantile functions	239
quasi-random sequences	235

R

R250 shift-register random number
 generator 229
 Racah coefficients 55
 Radial Mathieu Functions 81
 radioactivity, units of 483
 Radix-2 FFT for real data 195
 Radix-2 FFT, complex data 187
 rand, BSD random number generator
 227
 rand48 random number generator... 228
 random number distributions 239
 random number generators 217
 random sampling from histograms .. 306
 RANDU random number generator
 230
 RANF random number generator ... 228
 range 287
 range-checking for matrices 103
 range-checking for vectors 93
 RANLUX random number generator
 224
 RANLXD random number generator
 224
 RANLXS random number generator
 224
 RANMAR random number generator
 229, 230
 Rayleigh distribution 250
 Rayleigh Tail distribution 251
 real nonsymmetric matrix, eigensystem
 173
 real symmetric matrix, eigensystem
 171
 Reciprocal Gamma function 69
 rectangular to polar conversion 85
 recursive stratified sampling, MISER
 328
 reduction of angular variables 85
 refinement of solutions in linear systems
 154
 regression, least squares 443
 Regular Bessel Functions, Fractional
 Order 51
 Regular Bessel Functions, Zeros of... 52
 Regular Cylindrical Bessel Functions
 45
 Regular Modified Bessel Functions,
 Fractional Order 51

Regular Modified Cylindrical Bessel
 Functions 46
 Regular Modified Spherical Bessel
 Functions 49
 Regular Spherical Bessel Functions .. 48
 Regulated Gamma function 69
 relative Pochhammer symbol 71
 reporting bugs in GSL 5
 representations of complex numbers .. 27
 resampling from histograms 305
 residual, in nonlinear systems of
 equations 419, 459
 reversing a permutation 118
 Riemann Zeta Function 86
 RK2, Runge-Kutta method 349
 RK4, Runge-Kutta method 349
 RKF45, Runge-Kutta-Fehlberg method
 349
 root finding 389
 root finding, bisection algorithm 396
 root finding, Brent's method 396
 root finding, caveats 390
 root finding, false position algorithm
 396
 root finding, initial guess 394
 root finding, Newton's method 397
 root finding, overview 389
 root finding, providing a function to
 solve 391
 root finding, search bounds 394
 root finding, secant method 397
 root finding, Steffenson's method ... 398
 root finding, stopping parameters .. 395,
 419
 roots 389
 ROTG, Level-1 BLAS 141
 rounding mode 489
 Runge-Kutta Cash-Karp method ... 349
 Runge-Kutta methods, ordinary
 differential equations 349
 Runge-Kutta Prince-Dormand method
 349

S

safe comparison of floating point
 numbers 26
 sampling from histograms 305, 306
 SAXPY, Level-1 BLAS 140

SCAL, Level-1 BLAS	141	square root of a matrix, Cholesky	
schedule, cooling	337	decomposition	160
$sc(q, x)$, Mathieu function	81	square root of complex number	30
secant method for finding roots	397	standard deviation	287
selection function, ntuples	320	standard deviation, from histogram	
series, acceleration	375		303
shared libraries	8	standards conformance, ANSI C	7
shell prompt	6	Statistical Reference Datasets (StRD)	
$Shi(x)$	66		453
shift-register random number generator		statistics	287
	229	statistics, from histogram	303
$Si(x)$	66	steepest descent algorithm, minimization	
sign bit, IEEE format	487		435
sign of the determinant of a matrix		Steffenson's method for finding roots	
	155		398
simplex algorithm, minimization	436	stratified sampling in monte carlo	
simulated annealing	337	integration	325
sin, of complex number	30	stride, of vector index	91
sine function, special functions	84	Student t-distribution	262
single precision, IEEE format	487	subdiagonal, of a matrix	108
singular functions, numerical integration		summation, acceleration	375
of	211	superdiagonal, matrix	108
singular points, specifying positions in		SVD	159
quadrature	209	SWAP, Level-1 BLAS	140
singular value decomposition	159	swapping permutation elements	118
Skew Levy distribution	254	SYMM, Level-3 BLAS	146
skewness	289	symmetric matrix, real, eigensystem	
slope, see numerical derivative	367		171
Sobol sequence	235	SYMV, Level-2 BLAS	143
solution of linear system by Householder		synchrotron functions	83
transformations	166	SYR, Level-2 BLAS	144
solution of linear systems, $Ax=b$	153	SYR2, Level-2 BLAS	145
solving a nonlinear equation	389	SYR2K, Level-3 BLAS	149
solving nonlinear systems of equations		SYRK, Level-3 BLAS	148
	413	systems of equations, nonlinear	413
sorting	131		
sorting eigenvalues and eigenvectors			
	179		
sorting vector elements	132		
source code, reuse in applications	14		
special functions	41		
Spherical Bessel Functions	48		
spherical harmonics	75		
spherical random variates, 2D	266		
spherical random variates, 3D	266		
spherical random variates,			
N-dimensional	266		
spline	359		
splines, basis	469		

T

t-distribution	262
t-test	287
tangent of complex number	30
Tausworthe random number generator	
	225
Taylor coefficients, computation of	70
testing combination for validity	126
testing permutation for validity	118
thermal energy, units of	481
time units	479
trailing dimension, matrices	101

transformation, Householder	165
transforms, Hankel	387
transforms, wavelet	379
transport functions	83
traveling salesman problem	343
tridiagonal decomposition	161, 162
tridiagonal systems	166
trigonometric functions	84
trigonometric functions of complex numbers	30
trigonometric integrals	66
TRMM, Level-3 BLAS	147
TRMV, Level-2 BLAS	142
TRSM, Level-3 BLAS	147
TRSV, Level-2 BLAS	143
TSP	343
TT800 random number generator ...	229
two dimensional Gaussian distribution	245
two dimensional histograms	308
two-sided exponential distribution ..	247
Type 1 Gumbel distribution, random variates	268
Type 2 Gumbel distribution	269
U	
u-transform for series	375
underflow, IEEE exceptions	489
uniform distribution	257
units, conversion of	477
units, imperial	479
Unix random number generators, rand	227
Unix random number generators, rand48	227
unnormalized incomplete Gamma function	71
unweighted linear fits	443
usage, compiling application programs	7

V

value function, ntuples	320
Van der Pol oscillator, example	353
variance	287
variance, from histogram	303
variance-covariance matrix, linear fits	444
VAX random number generator	229
vector, operations	137
vector, sorting elements of	132
vectors	89, 91
vectors, initializing	94
vectors, range-checking	93
VEGAS monte carlo integration	330
viscosity, units of	482
volume units	480

W

W function	75
warning options	495
warranty (none)	5
wavelet transforms	379
website, developer information	6
Weibull distribution	267
weight, units of	481
weighted linear fits	443
Wigner coefficients	55

Y

y(x), Bessel Functions	49
Y(x), Bessel Functions	45

Z

zero finding	389
zero matrix	103
zero, IEEE format	487
Zeros of Regular Bessel Functions	52
Zeta functions	86
Ziggurat method	241