Package 'boostmath'

July 25, 2025
Title 'R' Bindings for the 'Boost' Math Functions
Version 1.0.0
Description 'R' bindings for the various functions and statistical distributions provided by the 'Boost' Math library https://www.boost.org/doc/libs/boost_1_88_0/libs/math/doc/html/index.html .
License MIT + file LICENSE
<pre>URL https://github.com/andrjohns/boostmath,</pre>
https://www.boost.org/doc/libs/boost_1_88_0/libs/math/doc/html/index.html
BugReports https://github.com/andrjohns/boostmath/issues
Encoding UTF-8
RoxygenNote 7.3.2
Depends R (>= $3.0.2$)
LinkingTo cpp11, BH (>= 1.87.0)
NeedsCompilation yes
Suggests knitr, rmarkdown, tinytest
VignetteBuilder knitr
Author Andrew R. Johnson [aut, cre] (ORCID: https://orcid.org/0000-0001-7000-8065)
Maintainer Andrew R. Johnson <andrew.johnson@arjohnsonau.com></andrew.johnson@arjohnsonau.com>
Repository CRAN
Date/Publication 2025-07-25 16:20:07 UTC
Contents
airy_functions

2 Contents

beta_distribution	 	9
beta_functions	 	10
binomial_distribution	 	12
cauchy_distribution	 	13
chebyshev_polynomials		
chi_squared_distribution		
double_exponential_quadrature		
elliptic_integrals		
error_functions		
exponential_distribution	 	20
exponential_integrals		
extreme_value_distribution		
factorials_and_binomial_coefficients		
fisher_f_distribution		
gamma_distribution		
gamma_functions		
gegenbauer_polynomials		
geometric_distribution		
hankel_functions	 	30
hermite_polynomials	 	30
holtsmark_distribution	 	31
hyperexponential_distribution		
hypergeometric_distribution		
hypergeometric_functions		
inverse_chi_squared_distribution		
inverse_gamma_distribution		
inverse_gaussian_distribution		
inverse_gaussian_distribution inverse_hyperbolic_functions		
jacobi_elliptic_functions		
jacobi_polynomials		
jacobi_theta_functions		
kolmogorov_smirnov_distribution		
laguerre_polynomials		
lambert_w_function		
landau_distribution		
laplace_distribution		
legendre_polynomials		
logistic_distribution		
lognormal_distribution		
mapairy_distribution		
negative_binomial_distribution		
non_central_beta_distribution		
non_central_chi_squared_distribution	 	54
non_central_t_distribution		
normal_distribution		
number_series	 	57
numerical_differentiation	 	58
numerical integration		59

airy_functions 3

	ooura_fourier_integrals	60
	owens_t	61
	pareto_distribution	62
	poisson_distribution	63
	polynomial_root_finding	64
	rayleigh_distribution	65
	rootfinding_and_minimisation	66
	saspoint5_distribution	68
	sinus_cardinal_hyperbolic_functions	69
	skew_normal_distribution	70
	spherical_harmonics	71
	students_t_distribution	72
	triangular_distribution	73
	uniform_distribution	74
	vector_functionals	75
	weibull_distribution	76
	zeta	77
Index		78

Description

Functions to compute the Airy functions Ai and Bi, their derivatives, and their zeros.

Usage

```
airy_ai(x)
airy_bi(x)
airy_ai_prime(x)
airy_bi_prime(x)
airy_ai_zero(m = NULL, start_index = NULL, number_of_zeros = NULL)
airy_bi_zero(m = NULL, start_index = NULL, number_of_zeros = NULL)
```

Arguments

```
x Input numeric value
m The index of the zero to find (1-based).
start_index The starting index for the zeros (1-based).
number_of_zeros
```

The number of zeros to find.

4 arcsine_distribution

Value

Single numeric value for the Airy functions and their derivatives, or a vector of length number_of_zeros for the multiple zero functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
airy_ai(2)
airy_bi(2)
airy_ai_prime(2)
airy_bi_prime(2)
airy_ai_zero(1)
airy_bi_zero(1)
airy_ai_zero(start_index = 1, number_of_zeros = 5)
airy_bi_zero(start_index = 1, number_of_zeros = 5)
```

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the arcsine distribution.

Usage

```
arcsine_pdf(x, x_min = 0, x_max = 1)
arcsine_lpdf(x, x_min = 0, x_max = 1)
arcsine_cdf(x, x_min = 0, x_max = 1)
arcsine_lcdf(x, x_min = 0, x_max = 1)
arcsine_quantile(p, x_min = 0, x_max = 1)
```

Arguments

```
    x quantile
    x_min minimum value of the distribution (default is 0)
    x_max maximum value of the distribution (default is 1)
    p probability
```

basic_functions 5

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
arcsine_pdf(0.5)
arcsine_lpdf(0.5)
arcsine_cdf(0.5)
arcsine_lcdf(0.5)
arcsine_quantile(0.5)
```

basic_functions

Basic Mathematical Functions

Description

Functions to compute sine, cosine, logarithm, exponential, cube root, square root, power, hypotenuse, and inverse square root.

Usage

```
sin_pi(x)
cos_pi(x)
log1p_boost(x)
expm1_boost(x)
cbrt(x)
sqrt1pm1(x)
powm1(x, y)
hypot(x, y)
rsqrt(x)
```

Arguments

Input numeric value

y Second input numeric value (for power and hypotenuse functions)

6 bernoulli_distribution

Value

A single numeric value with the computed result of the function.

See Also

Boost Documentation) for more details on the mathematical background.

Examples

```
# sin(pi * 0.5)
sin_pi(0.5)
# cos(pi * 0.5)
cos_pi(0.5)
# log(1 + 0.5)
log1p_boost(0.5)
# exp(0.5) - 1
expm1_boost(0.5)
cbrt(8)
# sqrt(1 + 0.5) - 1
sqrt1pm1(0.5)
# 2^3 - 1
powm1(2, 3)
hypot(3, 4)
rsqrt(4)
```

bernoulli_distribution

Bernoulli Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Bernoulli distribution.

```
bernoulli_pdf(x, p_success)
bernoulli_lpdf(x, p_success)
bernoulli_cdf(x, p_success)
bernoulli_lcdf(x, p_success)
bernoulli_quantile(p, p_success)
```

bessel_functions 7

Arguments

```
x quantile (0 or 1)
p_success probability of success (0 <= p_success <= 1)
probability (0 <= p <= 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
bernoulli_pdf(1, 0.5)
bernoulli_lpdf(1, 0.5)
bernoulli_cdf(1, 0.5)
bernoulli_lcdf(1, 0.5)
bernoulli_quantile(0.5, 0.5)
```

bessel_functions

Bessel Functions, Their Derivatives, and Zeros

Description

Functions to compute Bessel functions of the first and second kind, their modified versions, spherical Bessel functions, and their derivatives and zeros.

```
cyl_bessel_j(v, x)

cyl_neumann(v, x)

cyl_bessel_j_zero(v, m = NULL, start_index = NULL, number_of_zeros = NULL)

cyl_neumann_zero(v, m = NULL, start_index = NULL, number_of_zeros = NULL)

cyl_bessel_i(v, x)

cyl_bessel_k(v, x)

sph_bessel(v, x)
```

8 bessel_functions

```
cyl_bessel_j_prime(v, x)
cyl_neumann_prime(v, x)
cyl_bessel_i_prime(v, x)
cyl_bessel_k_prime(v, x)
sph_bessel_prime(v, x)
sph_neumann_prime(v, x)
```

Arguments

V	Order of the Bessel function
x	Argument of the Bessel function
m	The index of the zero to find (1-based).
start_index	The starting index for the zeros (1-based)
number_of_zeros	3
	The number of zeros to find.

Single numeric value for the Bessel functions and their derivatives, or a vector of length number_of_zeros for the multiple zero functions.

See Also

Value

Boost Documentation for more details on the mathematical background.

```
# Bessel function of the first kind J_0(1)
cyl_bessel_j(0, 1)
# Bessel function of the second kind Y_0(1)
cyl_neumann(0, 1)
# Modified Bessel function of the first kind I_0(1)
cyl_bessel_i(0, 1)
# Modified Bessel function of the second kind K_0(1)
cyl_bessel_k(0, 1)
# Spherical Bessel function of the first kind j_0(1)
sph_bessel(0, 1)
# Spherical Bessel function of the second kind y_0(1)
sph_neumann(0, 1)
# Derivative of the Bessel function of the first kind J_0(1)
cyl_bessel_j_prime(0, 1)
# Derivative of the Bessel function of the second kind Y_0(1)
cyl_neumann_prime(0, 1)
# Derivative of the modified Bessel function of the first kind I_{-}0(1)
```

beta_distribution 9

```
cyl_bessel_i_prime(0, 1)
# Derivative of the modified Bessel function of the second kind K_0(1)
cyl_bessel_k_prime(0, 1)
# Derivative of the spherical Bessel function of the first kind j_0(1)
sph_bessel_prime(0, 1)
# Derivative of the spherical Bessel function of the second kind y_0(1)
sph_neumann_prime(0, 1)
# Finding the first zero of the Bessel function of the first kind J_0
cyl_bessel_j_zero(0, 1)
# Finding the first zero of the Bessel function of the second kind Y_0
cyl_neumann_zero(0, 1)
# Finding multiple zeros of the Bessel function of the first kind J_0 starting from index 1
cyl_bessel_j_zero(0, start_index = 1, number_of_zeros = 5)
# Finding multiple zeros of the Bessel function of the second kind Y_0 starting from index 1
cyl_neumann_zero(0, start_index = 1, number_of_zeros = 5)
```

beta_distribution

Beta Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Beta distribution.

Usage

```
beta_pdf(x, alpha, beta)
beta_lpdf(x, alpha, beta)
beta_cdf(x, alpha, beta)
beta_lcdf(x, alpha, beta)
beta_quantile(p, alpha, beta)
```

Arguments

```
x quantile (0 \le x \le 1)
alpha shape parameter (alpha > 0)
beta shape parameter (beta > 0)
p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

beta_functions

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Beta distribution with shape parameters alpha = 2, beta = 5 beta_pdf(0.5, 2, 5) beta_lpdf(0.5, 2, 5) beta_cdf(0.5, 2, 5) beta_lcdf(0.5, 2, 5) beta_lcdf(0.5, 2, 5) beta_quantile(0.5, 2, 5)
```

beta_functions

Beta Functions

Description

Functions to compute the Euler beta function, normalised incomplete beta function, and their complements, as well as their inverses and derivatives.

```
beta_boost(a, b, x = NULL)
ibeta(a, b, x)
ibetac(a, b, x)
betac(a, b, x)
ibeta_inv(a, b, p)
ibeta_inva(b, x, p)
ibeta_inva(b, x, q)
ibeta_invb(a, x, p)
ibeta_invb(a, x, q)
ibeta_derivative(a, b, x)
```

beta_functions 11

Arguments

a	First parameter of the beta function
b	Second parameter of the beta function
x	Upper limit of integration $(0 \le x \le 1)$
p	Probability value $(0 \le p \le 1)$
q	Probability value $(0 \le q \le 1)$

Value

A single numeric value with the computed beta function, normalised incomplete beta function, or their complements, depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
## Not run:
# Euler beta function B(2, 3)
beta_boost(2, 3)
# Normalised incomplete beta function I_x(2, 3) for x = 0.5
ibeta(2, 3, 0.5)
# Normalised complement of the incomplete beta function 1 - I_x(2, 3) for x = 0.5
ibetac(2, 3, 0.5)
# Full incomplete beta function B_x(2, 3) for x = 0.5
beta_boost(2, 3, 0.5)
# Full complement of the incomplete beta function 1 - B_x(2, 3) for x = 0.5
betac(2, 3, 0.5)
# Inverse of the normalised incomplete beta function I_x(2, 3) = 0.5
ibeta_inv(2, 3, 0.5)
# Inverse of the normalised complement of the incomplete beta function I_x(2, 3) = 0.5
ibetac_inv(2, 3, 0.5)
# Inverse of the normalised complement of the incomplete beta function I_x(a, b)
# with respect to a for x = 0.5 and q = 0.5
ibetac_inva(3, 0.5, 0.5)
# Inverse of the normalised incomplete beta function I_x(a, b)
# with respect to b for x = 0.5 and p = 0.5
ibeta_invb(0.8, 0.5, 0.5)
# Inverse of the normalised complement of the incomplete beta function I_x(a, b)
# with respect to b for x = 0.5 and q = 0.5
ibetac_invb(2, 0.5, 0.5)
# Derivative of the incomplete beta function with respect to x for a = 2, b = 3, x = 0.5
ibeta_derivative(2, 3, 0.5)
## End(Not run)
```

12 binomial_distribution

binomial_distribution Binomial Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Binomial distribution.

Usage

```
binomial_pdf(k, n, prob)
binomial_lpdf(k, n, prob)
binomial_cdf(k, n, prob)
binomial_lcdf(k, n, prob)
binomial_quantile(p, n, prob)
```

Arguments

```
k number of successes (0 \le k \le n)

n number of trials (n \ge 0)

probability of success on each trial (0 \le prob \le 1)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Binomial dist ribution with n = 10, prob = 0.5
binomial_pdf(3, 10, 0.5)
binomial_lpdf(3, 10, 0.5)
binomial_cdf(3, 10, 0.5)
binomial_lcdf(3, 10, 0.5)
binomial_quantile(0.5, 10, 0.5)
```

cauchy_distribution 13

cauchy_distribution Cauchy D

Cauchy Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Cauchy distribution.

Usage

```
cauchy_pdf(x, location = 0, scale = 1)
cauchy_lpdf(x, location = 0, scale = 1)
cauchy_cdf(x, location = 0, scale = 1)
cauchy_lcdf(x, location = 0, scale = 1)
cauchy_quantile(p, location = 0, scale = 1)
```

Arguments

```
x quantile location location parameter (default is 0) scale scale parameter (default is 1) p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Cauchy distribution with location = 0, scale = 1
cauchy_pdf(0)
cauchy_lpdf(0)
cauchy_cdf(0)
cauchy_lcdf(0)
cauchy_quantile(0.5)
```

chebyshev_polynomials Chebyshev Polynomials and Related Functions

Description

Functions to compute Chebyshev polynomials of the first and second kind.

Usage

```
chebyshev_next(x, Tn, Tn_1)
chebyshev_t(n, x)
chebyshev_u(n, x)
chebyshev_t_prime(n, x)
chebyshev_clenshaw_recurrence(c, x)
chebyshev_clenshaw_recurrence_ab(c, a, b, x)
```

Arguments

X	Argument of the polynomial
Tn	Value of the Chebyshev polynomial $(T_n(x))$
Tn_1	Value of the Chebyshev polynomial $(T_{n-1}(x))$
n	Degree of the polynomial
С	Coefficients of the Chebyshev polynomial
а	Lower bound of the interval
b	Upper bound of the interval

Value

A single numeric value with the computed Chebyshev polynomial, its derivative, or related functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Chebyshev polynomial of the first kind T_2(0.5) chebyshev_t(2, 0.5)
# Chebyshev polynomial of the second kind U_2(0.5) chebyshev_u(2, 0.5)
# Derivative of the Chebyshev polynomial of the first kind T_2'(0.5) chebyshev_t_prime(2, 0.5)
# Next Chebyshev polynomial of the first kind T_3(0.5) using T_2(0.5) and T_1(0.5) chebyshev_next(0.5, chebyshev_t(2, 0.5), chebyshev_t(1, 0.5))
# Chebyshev polynomial of the first kind using Clenshaw's recurrence with coefficients
# C = C(1, 0, -1) at C = C(1,
```

chi_squared_distribution

Chi-Squared Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Chi-Squared distribution.

Usage

```
chi_squared_pdf(x, df)
chi_squared_lpdf(x, df)
chi_squared_cdf(x, df)
chi_squared_lcdf(x, df)
chi_squared_lcdf(x, df)
```

Arguments

```
x quantile

df degrees of freedom (df > 0)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Chi-Squared distribution with 3 degrees of freedom
chi_squared_pdf(2, 3)
chi_squared_lpdf(2, 3)
chi_squared_cdf(2, 3)
chi_squared_lcdf(2, 3)
chi_squared_quantile(0.5, 3)
```

double_exponential_quadrature

Double Exponential Quadrature

Description

Functions for numerical integration using double exponential quadrature methods such as tanh-sinh, sinh-sinh, and exp-sinh quadrature.

Usage

```
tanh_sinh(f, a, b, tol = sqrt(.Machine$double.eps), max_refinements = 15)
sinh_sinh(f, tol = sqrt(.Machine$double.eps), max_refinements = 9)
exp_sinh(f, a, b, tol = sqrt(.Machine$double.eps), max_refinements = 9)
```

Arguments

A function to integrate. It should accept a single numeric value and return a single numeric value.

a The lower limit of integration.

b The upper limit of integration.

tol The tolerance for the approximation. Default is sqrt(.Machine\$double.eps).

max_refinements

The maximum number of refinements to apply. Default is 15 for tanh-sinh and 9 for sinh-sinh and exp-sinh.

Value

A single numeric value with the computed integral.

elliptic_integrals 17

Examples

```
# Tanh-sinh quadrature of log(x) from 0 to 1 tanh_sinh(function(x) { log(x) * log1p(-x) }, a = 0, b = 1) # Sinh-sinh quadrature of exp(-x^2) sinh_sinh(function(x) { exp(-x * x) }) # Exp-sinh quadrature of exp(-3*x) from 0 to Inf exp_sinh(function(x) { <math>exp(-3 * x) }, a = 0, b = Inf)
```

elliptic_integrals

Elliptic Integrals

Description

Functions to compute various elliptic integrals, including Carlson's elliptic integrals and incomplete elliptic integrals.

Usage

```
ellint_rf(x, y, z)
ellint_rd(x, y, z)
ellint_rj(x, y, z, p)
ellint_rc(x, y)
ellint_rg(x, y, z)
ellint_1(k, phi = NULL)
ellint_2(k, phi = NULL)
ellint_3(k, n, phi = NULL)
ellint_d(k, phi = NULL)
jacobi_zeta(k, phi)
heuman_lambda(k, phi)
```

Arguments

Χ	First parameter of the integral
у	Second parameter of the integral
Z	Third parameter of the integral
р	Fourth parameter of the integral (for Ri)

18 elliptic_integrals

k	Elliptic modulus (for incomplete elliptic integrals)
phi	Amplitude (for incomplete elliptic integrals)
n	Characteristic (for incomplete elliptic integrals of the third kind)

Value

A single numeric value with the computed elliptic integral.

See Also

Boost Documentation for more details on the mathematical background.

```
# Carlson's elliptic integral Rf with parameters x = 1, y = 2, z = 3
ellint_rf(1, 2, 3)
\#' # Carlson's elliptic integral Rd with parameters x = 1, y = 2, z = 3
ellint_rd(1, 2, 3)
# Carlson's elliptic integral Rj with parameters x = 1, y = 2, z = 3, p = 4
ellint_rj(1, 2, 3, 4)
# Carlson's elliptic integral Rc with parameters x = 1, y = 2
ellint_rc(1, 2)
# Carlson's elliptic integral Rg with parameters x = 1, y = 2, z = 3
ellint_rg(1, 2, 3)
# Incomplete elliptic integral of the first kind with k = 0.5, phi = pi/4
ellint_1(0.5, pi / 4)
# Complete elliptic integral of the first kind
ellint_1(0.5)
# Incomplete elliptic integral of the second kind with k = 0.5, phi = pi/4
ellint_2(0.5, pi / 4)
# Complete elliptic integral of the second kind
ellint_2(0.5)
# Incomplete elliptic integral of the third kind with k = 0.5, n = 0.5, phi = pi/4
ellint_3(0.5, 0.5, pi / 4)
# Complete elliptic integral of the third kind with k = 0.5, n = 0.5
ellint_3(0.5, 0.5)
# Incomplete elliptic integral D with k = 0.5, phi = pi/4
ellint_d(0.5, pi / 4)
# Complete elliptic integral D
ellint_d(0.5)
# Jacobi zeta function with k = 0.5, phi = pi/4
jacobi_zeta(0.5, pi / 4)
# Heuman's lambda function with k = 0.5, phi = pi/4
heuman_lambda(0.5, pi / 4)
```

error_functions 19

error_functions

Error Functions and Inverses

Description

Functions to compute the error function, complementary error function, and their inverses.

Usage

```
erf(x)
erfc(x)
erf_inv(p)
erfc_inv(p)
```

Arguments

x Input numeric value

p Probability value $(0 \le p \le 1)$

Value

A single numeric value with the computed error function, complementary error function, or their inverses.

See Also

Boost Documentation for more details

```
# Error function
erf(0.5)
# Complementary error function
erfc(0.5)
# Inverse error function
erf_inv(0.5)
# Inverse complementary error function
erfc_inv(0.5)
```

```
exponential_distribution
```

Exponential Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Exponential distribution.

Usage

```
exponential_pdf(x, lambda)
exponential_lpdf(x, lambda)
exponential_cdf(x, lambda)
exponential_lcdf(x, lambda)
exponential_quantile(p, lambda)
```

Arguments

```
x quantile

lambda rate parameter (lambda > 0)

p probability (0 <= p <= 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Exponential distribution with rate parameter lambda = 2
exponential_pdf(1, 2)
exponential_lpdf(1, 2)
exponential_cdf(1, 2)
exponential_lcdf(1, 2)
exponential_quantile(0.5, 2)
```

exponential_integrals 21

Description

Functions to compute various exponential integrals, including En and Ei.

Usage

```
expint_en(n, z)
expint_ei(z)
```

Arguments

- n Order of the integral (for En)
- z Argument of the integral (for En and Ei)

Value

A single numeric value with the computed exponential integral.

See Also

Boost Documentation for

Examples

```
# Exponential integral En with n = 1 and z = 2 expint_en(1, 2) # Exponential integral Ei with z = 2 expint_ei(2)
```

extreme_value_distribution

Extreme Value Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Extreme Value distribution.

Usage

```
extreme_value_pdf(x, location = 0, scale = 1)
extreme_value_lpdf(x, location = 0, scale = 1)
extreme_value_cdf(x, location = 0, scale = 1)
extreme_value_lcdf(x, location = 0, scale = 1)
extreme_value_quantile(p, location = 0, scale = 1)
```

Arguments

```
x quantile
location location parameter (default is 0)
scale scale parameter (default is 1)
p probability (0 <= p <= 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Extreme Value distribution with location = 0, scale = 1
extreme_value_pdf(0)
extreme_value_lpdf(0)
extreme_value_cdf(0)
extreme_value_lcdf(0)
extreme_value_quantile(0.5)
```

```
factorials_and_binomial_coefficients

Factorials and Binomial Coefficients
```

Description

Functions to compute factorials, double factorials, rising and falling factorials, and binomial coefficients.

Usage

```
factorial_boost(i)
unchecked_factorial(i)
max_factorial()
double_factorial(i)
rising_factorial(x, i)
falling_factorial(x, i)
binomial_coefficient(n, k)
```

Arguments

- i Non-negative integer input for factorials and double factorials.
- x Base value for rising and falling factorials.
- n Total number of elements for binomial coefficients.
- k Number of elements to choose for binomial coefficients.

Value

A single numeric value with the computed factorial, double factorial, rising factorial, falling factorial, or binomial coefficient.

See Also

Boost Documentation for more details on the mathematical background.

```
# Factorial of 5
factorial_boost(5)
# Unchecked factorial of 5 (using table lookup)
unchecked_factorial(5)
# Maximum factorial value that can be computed
max_factorial()
# Double factorial of 6
double_factorial(6)
# Rising factorial of 3 with exponent 2
rising_factorial(3, 2)
# Falling factorial of 3 with exponent 2
falling_factorial(3, 2)
# Binomial coefficient "5 choose 2"
binomial_coefficient(5, 2)
```

24 fisher_f_distribution

```
fisher_f_distribution Fisher F Distribution Functions
```

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Fisher F distribution.

Usage

```
fisher_f_pdf(x, df1, df2)
fisher_f_lpdf(x, df1, df2)
fisher_f_cdf(x, df1, df2)
fisher_f_lcdf(x, df1, df2)
fisher_f_quantile(p, df1, df2)
```

Arguments

```
x quantile

df1 degrees of freedom for the numerator (df1 > 0)

df2 degrees of freedom for the denominator (df2 > 0)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Fisher F distribution with df1 = 5, df2 = 2
fisher_f_pdf(1, 5, 2)
fisher_f_lpdf(1, 5, 2)
fisher_f_cdf(1, 5, 2)
fisher_f_lcdf(1, 5, 2)
fisher_f_quantile(0.5, 5, 2)
```

gamma_distribution 25

gamma_distribution Gam

Gamma Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Gamma distribution.

Usage

```
gamma_pdf(x, shape, scale)
gamma_lpdf(x, shape, scale)
gamma_cdf(x, shape, scale)
gamma_lcdf(x, shape, scale)
gamma_quantile(p, shape, scale)
```

Arguments

```
x quantile

shape shape parameter (shape > 0)

scale scale parameter (scale > 0)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Gamma distribution with shape = 3, scale = 4
gamma_pdf(2, 3, 4)
gamma_lpdf(2, 3, 4)
gamma_lcdf(2, 3, 4)
gamma_lcdf(2, 3, 4)
gamma_quantile(0.5, 3, 4)
```

26 gamma_functions

gamma_functions

Gamma Functions

Description

Functions to compute the gamma function, its logarithm, digamma, trigamma, polygamma, and various incomplete gamma functions.

Usage

```
tgamma(z)
tgamma1pm1(z)
lgamma_boost(z)
digamma_boost(z)
trigamma_boost(z)
polygamma(n, z)
tgamma_ratio(a, b)
tgamma_delta_ratio(a, delta)
gamma_p(a, z)
gamma_q(a, z)
tgamma_lower(a, z)
tgamma_upper(a, z)
gamma_q_inv(a, q)
gamma_p_inv(a, p)
gamma_q_inva(z, q)
gamma_p_inva(z, p)
gamma_p_derivative(a, z)
```

Arguments

Input numeric value for the gamma function

gamma_functions 27

n	Order of the polygamma function (non-negative integer)
а	Argument for the incomplete gamma functions
b	Denominator argument for the ratio of gamma functions
delta	Increment for the ratio of gamma functions
q	Probability value for the incomplete gamma functions
р	Probability value for the incomplete gamma functions

Value

A single numeric value with the computed gamma function, logarithm, digamma, trigamma, polygamma, or incomplete gamma functions.

See Also

Boost Documentation for more details on the mathematical background.

```
## Not run:
# Gamma function for z = 5
tgamma(5)
# Gamma function for (1 + z) - 1, where z = 5
tgamma1pm1(5)
\# Logarithm of the gamma function for z = 5
lgamma_boost(5)
# Digamma function for z = 5
digamma_boost(5)
# Trigamma function for z = 5
trigamma_boost(5)
# Polygamma function of order 1 for z = 5
polygamma(1, 5)
# Ratio of gamma functions for a = 5, b = 3
tgamma_ratio(5, 3)
# Ratio of gamma functions with delta for a = 5, delta = 2
tgamma_delta_ratio(5, 2)
# Normalised lower incomplete gamma function P(a, z) for a = 5, z = 2
gamma_p(5, 2)
# Normalised upper incomplete gamma function Q(a, z) for a = 5, z = 2
gamma_q(5, 2)
# Full lower incomplete gamma function for a = 5, z = 2
tgamma_lower(5, 2)
# Full upper incomplete gamma function for a = 5, z = 2
tgamma_upper(5, 2)
# Inverse of the normalised upper incomplete gamma function for a = 5, q = 0.5
gamma_q_inv(5, 0.5)
# Inverse of the normalised lower incomplete gamma function for a = 5, p = 0.5
gamma_p_inv(5, 0.5)
# Inverse of the normalised upper incomplete gamma function with respect to a for z = 2, q = 0.5
gamma_q_inva(2, 0.5)
# Inverse of the normalised lower incomplete gamma function with respect to a for z = 2, p = 0.5
```

```
gamma_p_inva(2, 0.5)
# Derivative of the normalised upper incomplete gamma function for a = 5, z = 2
gamma_p_derivative(5, 2)
## End(Not run)
```

gegenbauer_polynomials

Gegenbauer Polynomials and Related Functions

Description

Functions to compute Gegenbauer polynomials, their derivatives, and related functions.

Usage

```
gegenbauer(n, lambda, x)
gegenbauer_prime(n, lambda, x)
gegenbauer_derivative(n, lambda, x, k)
```

Arguments

n	Degree of the polynomial
lambda	Parameter of the polynomial
x	Argument of the polynomial
k	Order of the derivative

Value

A single numeric value with the computed Gegenbauer polynomial, its derivative, or k-th derivative.

See Also

Boost Documentation for more details on the mathematical background.

```
# Gegenbauer polynomial C_2^{(1)}(0.5) gegenbauer(2, 1, 0.5) 
# Derivative of the Gegenbauer polynomial C_2^{(1)}(0.5) gegenbauer_prime(2, 1, 0.5) 
# k-th derivative of the Gegenbauer polynomial C_2^{(1)}(0.5) gegenbauer_derivative(2, 1, 0.5, 2)
```

geometric_distribution 29

```
geometric_distribution
```

Geometric Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Geometric distribution.

Usage

```
geometric_pdf(x, prob)
geometric_lpdf(x, prob)
geometric_cdf(x, prob)
geometric_lcdf(x, prob)
geometric_quantile(p, prob)
```

Arguments

```
x quantile (non-negative integer)
prob probability of success (0 < prob < 1)
probability (0 <= p <= 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Geometric distribution with probability of success prob = 0.5
geometric_pdf(3, 0.5)
geometric_lpdf(3, 0.5)
geometric_cdf(3, 0.5)
geometric_lcdf(3, 0.5)
geometric_quantile(0.5, 0.5)
```

30 hermite_polynomials

hankel_functions

Hankel Functions

Description

Functions to compute cyclic and spherical Hankel functions of the first and second kinds.

Usage

```
cyl_hankel_1(v, x)
cyl_hankel_2(v, x)
sph_hankel_1(v, x)
sph_hankel_2(v, x)
```

Arguments

v Order of the Hankel function

x Argument of the Hankel function

Value

A single complex value with the computed Hankel function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
cyl_hankel_1(2, 0.5)
cyl_hankel_2(2, 0.5)
sph_hankel_1(2, 0.5)
sph_hankel_2(2, 0.5)
```

hermite_polynomials

Hermite Polynomials and Related Functions

Description

Functions to compute Hermite polynomials.

holtsmark_distribution 31

Usage

```
hermite(n, x)
hermite_next(n, x, Hn, Hnm1)
```

Arguments

n	Degree of the polynomial
x	Argument of the polynomial
Hn	Value of the Hermite polynomial $(H_n(x))$
Hnm1	Value of the Hermite polynomial $(H_{n-1}(x))$

Value

A single numeric value with the computed Hermite polynomial or its next value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hermite polynomial H_2(0.5) hermite(2, 0.5) # Next Hermite polynomial H_3(0.5) using H_2(0.5) and H_1(0.5) hermite_next(2, 0.5, hermite(2, 0.5), hermite(1, 0.5))
```

holtsmark_distribution

Holtsmark Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Holtsmark distribution.

```
holtsmark_pdf(x, location = 0, scale = 1)
holtsmark_lpdf(x, location = 0, scale = 1)
holtsmark_cdf(x, location = 0, scale = 1)
holtsmark_lcdf(x, location = 0, scale = 1)
holtsmark_quantile(p, location = 0, scale = 1)
```

X	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability $(0 \le p \le 1)$

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Distribution only available with Boost version 1.87.0 or later.
## Not run:
# Holtsmark distribution with location 0 and scale 1
holtsmark_pdf(3)
holtsmark_lpdf(3)
holtsmark_cdf(3)
holtsmark_lcdf(3)
holtsmark_lcdf(3)
## End(Not run)
```

hyperexponential_distribution

Hyperexponential Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Hyperexponential distribution.

```
hyperexponential_pdf(x, probabilities, rates)
hyperexponential_lpdf(x, probabilities, rates)
hyperexponential_cdf(x, probabilities, rates)
hyperexponential_lcdf(x, probabilities, rates)
hyperexponential_quantile(p, probabilities, rates)
```

```
x quantile probabilities vector of probabilities (sum must be 1) rates vector of rates (all rates must be > 0) probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hyperexponential distribution with probabilities = c(0.5, 0.5) and rates = c(1, 2) hyperexponential_pdf(2, c(0.5, 0.5), c(1, 2)) hyperexponential_lpdf(2, c(0.5, 0.5), c(1, 2)) hyperexponential_cdf(2, c(0.5, 0.5), c(1, 2)) hyperexponential_lcdf(2, c(0.5, 0.5), c(1, 2)) hyperexponential_quantile(0.5, c(0.5, 0.5), c(1, 2))
```

hypergeometric_distribution

Hypergeometric Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Hypergeometric distribution.

```
hypergeometric_pdf(x, r, n, N)
hypergeometric_lpdf(x, r, n, N)
hypergeometric_cdf(x, r, n, N)
hypergeometric_lcdf(x, r, n, N)
hypergeometric_quantile(p, r, n, N)
```

```
x quantile (non-negative integer)
r number of successes in the population (r >= 0)
n number of draws (n >= 0)
N population size (N >= r)
p probability (0 <= p <= 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hypergeometric distribution with r = 5, n = 10, N = 20 hypergeometric_pdf(3, 5, 10, 20) hypergeometric_lpdf(3, 5, 10, 20) hypergeometric_cdf(3, 5, 10, 20) hypergeometric_lcdf(3, 5, 10, 20) hypergeometric_quantile(0.5, 5, 10, 20)
```

hypergeometric_functions

Hypergeometric Functions

Description

Functions to compute various hypergeometric functions.

```
hypergeometric_1F0(a, z)
hypergeometric_0F1(b, z)
hypergeometric_2F0(a1, a2, z)
hypergeometric_1F1(a, b, z)
hypergeometric_1F1_regularized(a, b, z)
log_hypergeometric_1F1(a, b, z)
hypergeometric_pFq(a, b, z)
```

a	Parameter of the hypergeometric function
z	Argument of the hypergeometric function
b	Second parameter of the hypergeometric function
a1	First parameter of the hypergeometric function
a2	Second parameter of the hypergeometric function

Value

A single numeric value with the computed hypergeometric function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hypergeometric Function 1F0
hypergeometric_1F0(2, 0.2)
# Hypergeometric Function 0F1
hypergeometric_0F1(1, 0.8)
# Hypergeometric Function 2F0
hypergeometric_2F0(0.1, -1, 0.1)
# Hypergeometric Function 1F1
hypergeometric_1F1(2, 3, 1)
# Regularised Hypergeometric Function 1F1
hypergeometric_1F1_regularized(2, 3, 1)
# Logarithm of the Hypergeometric Function 1F1
log_hypergeometric_1F1(2, 3, 1)
# Hypergeometric_Function pFq
hypergeometric_pFq(c(2, 3), c(4, 5), 6)
```

```
inverse\_chi\_squared\_distribution\\ Inverse\ Chi-Squared\ Distribution\ Functions
```

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Inverse Chi-Squared distribution.

Usage

```
inverse_chi_squared_pdf(x, df, scale = 1)
inverse_chi_squared_lpdf(x, df, scale = 1)
inverse_chi_squared_cdf(x, df, scale = 1)
inverse_chi_squared_lcdf(x, df, scale = 1)
inverse_chi_squared_quantile(p, df, scale = 1)
```

Arguments

```
x quantile

df degrees of freedom (df > 0)

scale scale parameter (default is 1)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Chi-Squared distribution with 3 degrees of freedom, scale = 1
inverse_chi_squared_pdf(2, 3, 1)
inverse_chi_squared_lpdf(2, 3, 1)
inverse_chi_squared_cdf(2, 3, 1)
inverse_chi_squared_lcdf(2, 3, 1)
inverse_chi_squared_quantile(0.5, 3, 1)
```

```
inverse\_gamma\_distribution
```

Inverse Gamma Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Inverse Gamma distribution.

Usage

```
inverse_gamma_pdf(x, shape, scale)
inverse_gamma_lpdf(x, shape, scale)
inverse_gamma_cdf(x, shape, scale)
inverse_gamma_lcdf(x, shape, scale)
inverse_gamma_quantile(p, shape, scale)
```

Arguments

X	quantile
shape	shape parameter (shape > 0)
scale	scale parameter (scale > 0)
р	probability $(0 \le p \le 1)$

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Gamma distribution with shape = 3, scale = 4
inverse_gamma_pdf(2, 3, 4)
inverse_gamma_lpdf(2, 3, 4)
inverse_gamma_cdf(2, 3, 4)
inverse_gamma_lcdf(2, 3, 4)
inverse_gamma_quantile(0.5, 3, 4)
```

```
inverse\_gaussian\_distribution
```

Inverse Gaussian Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Inverse Gaussian distribution.

Usage

```
inverse_gaussian_pdf(x, mu, lambda)
inverse_gaussian_lpdf(x, mu, lambda)
inverse_gaussian_cdf(x, mu, lambda)
inverse_gaussian_lcdf(x, mu, lambda)
inverse_gaussian_quantile(p, mu, lambda)
```

Arguments

```
x quantile

mu mean parameter (mu > 0)

lambda scale (precision) parameter (lambda > 0)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Gaussian distribution with mu = 3, lambda = 4
inverse_gaussian_pdf(2, 3, 4)
inverse_gaussian_lpdf(2, 3, 4)
inverse_gaussian_cdf(2, 3, 4)
inverse_gaussian_lcdf(2, 3, 4)
inverse_gaussian_quantile(0.5, 3, 4)
```

```
inverse\_hyperbolic\_functions
```

Inverse Hyperbolic Functions

Description

Functions to compute the inverse hyperbolic functions: acosh, asinh, and atanh.

Usage

```
acosh_boost(x)
asinh_boost(x)
atanh_boost(x)
```

Arguments

Χ

Input numeric value

Value

A single numeric value with the computed inverse hyperbolic function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Hyperbolic Cosine
acosh_boost(2)
# Inverse Hyperbolic Sine
asinh_boost(1)
# Inverse Hyperbolic Tangent
atanh_boost(0.5)
```

```
jacobi_elliptic_functions
```

Jacobi Elliptic Functions

Description

Functions to compute the Jacobi elliptic functions: sn, cn, dn, and others.

Usage

```
jacobi_elliptic(k, u)
jacobi_cd(k, u)
jacobi_cn(k, u)
jacobi_cs(k, u)
jacobi_dc(k, u)
```

```
jacobi_dn(k, u)
jacobi_ds(k, u)
jacobi_nc(k, u)
jacobi_nd(k, u)
jacobi_ns(k, u)
jacobi_sc(k, u)
jacobi_sd(k, u)
jacobi_sn(k, u)
```

Arguments

```
k Elliptic modulus (0 \le k \le 1)
u Argument of the elliptic functions
```

Value

For jacobi_elliptic, a list containing the values of the Jacobi elliptic functions: sn, cn, dn. For individual functions, a single numeric value is returned.

See Also

Boost Documentation for more details on the mathematical background.

```
# Jacobi Elliptic Functions
k < -0.5
u <- 2
jacobi_elliptic(k, u)
# Individual Jacobi Elliptic Functions
jacobi_cd(k, u)
jacobi_cn(k, u)
jacobi_cs(k, u)
jacobi_dc(k, u)
jacobi_dn(k, u)
jacobi_ds(k, u)
jacobi_nc(k, u)
jacobi_nd(k, u)
jacobi_ns(k, u)
jacobi_sc(k, u)
jacobi_sd(k, u)
jacobi_sn(k, u)
```

jacobi_polynomials 41

Description

Functions to compute Jacobi polynomials, their derivatives, and related functions.

Usage

```
jacobi(n, alpha, beta, x)
jacobi_prime(n, alpha, beta, x)
jacobi_double_prime(n, alpha, beta, x)
jacobi_derivative(n, alpha, beta, x, k)
```

Arguments

n	Degree of the polynomial
alpha	First parameter of the polynomial
beta	Second parameter of the polynomial
x	Argument of the polynomial
k	Order of the derivative

Value

A single numeric value with the computed Jacobi polynomial, its derivative, or k-th derivative.

See Also

Boost Documentation for more details on the mathematical background.

```
# Jacobi polynomial P_2^(1, 2)(0.5)
jacobi(2, 1, 2, 0.5)
# Derivative of the Jacobi polynomial P_2^(1, 2)'(0.5)
jacobi_prime(2, 1, 2, 0.5)
# Second derivative of the Jacobi polynomial P_2^(1, 2)''(0.5)
jacobi_double_prime(2, 1, 2, 0.5)
# 3rd derivative of the Jacobi polynomial P_2^(1, 2)^(k)(0.5)
jacobi_derivative(2, 1, 2, 0.5, 3)
```

```
jacobi_theta_functions
```

Jacobi Theta Functions

Description

Functions to compute the Jacobi theta functions $(\theta_1, \theta_2, \theta_3, \theta_4)$ parameterised by either (q) or (τ) .

Usage

```
jacobi_theta1(x, q)

jacobi_theta1tau(x, tau)

jacobi_theta2(x, q)

jacobi_theta2tau(x, tau)

jacobi_theta3(x, q)

jacobi_theta3tau(x, tau)

jacobi_theta3m1(x, q)

jacobi_theta3m1tau(x, tau)

jacobi_theta4(x, q)

jacobi_theta4tau(x, tau)

jacobi_theta4tau(x, tau)

jacobi_theta4tau(x, tau)

jacobi_theta4tau(x, tau)
```

Arguments

x Input value

q The nome parameter of the Jacobi theta function (0 < q < 1)

The nome parameter of the Jacobi theta function (tau = u + iv, where u and v are

real numbers)

Value

A single numeric value with the computed Jacobi theta function.

See Also

Examples

```
# Jacobi Theta Functions
x < -0.5
q < -0.9
tau <- 1.5
jacobi_theta1(x, q)
jacobi_theta1tau(x, tau)
jacobi_theta2(x, q)
jacobi_theta2tau(x, tau)
jacobi_theta3(x, q)
jacobi_theta3tau(x, tau)
jacobi_theta3m1(x, q)
jacobi_theta3m1tau(x, tau)
jacobi_theta4(x, q)
jacobi_theta4tau(x, tau)
jacobi_theta4m1(x, q)
jacobi_theta4m1tau(x, tau)
```

kolmogorov_smirnov_distribution

Kolmogorov-Smirnov Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Kolmogorov-Smirnov distribution.

Usage

```
kolmogorov_smirnov_pdf(x, n)
kolmogorov_smirnov_lpdf(x, n)
kolmogorov_smirnov_cdf(x, n)
kolmogorov_smirnov_lcdf(x, n)
kolmogorov_smirnov_quantile(p, n)
```

Arguments

```
x quantile

n sample size (n > 0)

p probability (0 \le p \le 1)
```

Value

Boost Documentation for more details on the mathematical background.

Examples

```
# Kolmogorov-Smirnov distribution with sample size n = 10
kolmogorov_smirnov_pdf(0.5, 10)
kolmogorov_smirnov_lpdf(0.5, 10)
kolmogorov_smirnov_cdf(0.5, 10)
kolmogorov_smirnov_lcdf(0.5, 10)
kolmogorov_smirnov_quantile(0.5, 10)
```

Description

Functions to compute Laguerre polynomials of the first kind.

Usage

```
laguerre(n, x)
laguerre_m(n, m, x)
laguerre_next(n, x, Ln, Lnm1)
laguerre_next_m(n, m, x, Ln, Lnm1)
```

Arguments

n	Degree of the polynomial
X	Argument of the polynomial
m	Order of the polynomial (for Laguerre polynomials of the first kind)
Ln	Value of the Laguerre polynomial $(L_n(x))$
Lnm1	Value of the Laguerre polynomial $(L_{n-1}(x))$

Value

A single numeric value with the computed Laguerre polynomial, its derivative, or related functions.

See Also

lambert_w_function 45

Examples

```
# Laguerre polynomial of the first kind L_2(0.5)
laguerre(2, 0.5)
# Laguerre polynomial of the first kind with order 1 L_2^1(0.5)
laguerre_m(2, 1, 0.5)
# Next Laguerre polynomial of the first kind L_3(0.5) using L_2(0.5) and L_1(0.5)
laguerre_next(2, 0.5, laguerre(2, 0.5), laguerre(1, 0.5))
# Next Laguerre polynomial of the first kind with order 1 L_3^1(0.5) using L_2^1(0.5) and L_1^1(0.5)
laguerre_next_m(2, 1, 0.5, laguerre_m(2, 1, 0.5), laguerre_m(1, 1, 0.5))
```

lambert_w_function

Lambert W Function and Its Derivatives

Description

Functions to compute the Lambert W function and its derivatives for the principal branch (W_0) and the branch -1 (W_{-1}) .

Usage

```
lambert_w0(z)
lambert_wm1(z)
lambert_w0_prime(z)
lambert_wm1_prime(z)
```

Arguments

Z

Argument of the Lambert W function

Value

A single numeric value with the computed Lambert W function or its derivative.

See Also

Boost Documentation for more details on the mathematical background.

```
# Lambert W Function (Principal Branch)
lambert_w0(0.3)
# Lambert W Function (Branch -1)
lambert_wm1(-0.3)
# Derivative of the Lambert W Function (Principal Branch)
lambert_w0_prime(0.3)
# Derivative of the Lambert W Function (Branch -1)
lambert_wm1_prime(-0.3)
```

46 landau_distribution

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Landau distribution.

Usage

```
landau_pdf(x, location = 0, scale = 1)
landau_lpdf(x, location = 0, scale = 1)
landau_cdf(x, location = 0, scale = 1)
landau_lcdf(x, location = 0, scale = 1)
landau_quantile(p, location = 0, scale = 1)
```

Arguments

```
x quantile location location parameter (default is 0) scale scale parameter (default is 1) p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Distribution only available with Boost version 1.87.0 or later.
## Not run:
# Landau distribution with location 0 and scale 1
   landau_pdf(3)
   landau_lpdf(3)
   landau_cdf(3)
   landau_lcdf(3)
   landau_quantile(0.5)
## End(Not run)
```

laplace_distribution 47

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Laplace distribution.

Usage

```
laplace_pdf(x, location = 0, scale = 1)
laplace_lpdf(x, location = 0, scale = 1)
laplace_cdf(x, location = 0, scale = 1)
laplace_lcdf(x, location = 0, scale = 1)
laplace_quantile(p, location = 0, scale = 1)
```

Arguments

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Laplace distribution with location = 0, scale = 1
laplace_pdf(0)
laplace_lpdf(0)
laplace_cdf(0)
laplace_lcdf(0)
laplace_quantile(0.5)
```

Description

Functions to compute Legendre polynomials of the first and second kind, their derivatives, zeros, and related functions.

Usage

```
legendre_p(n, x)
legendre_p_prime(n, x)
legendre_p_zeros(n)
legendre_p_m(n, m, x)
legendre_q(n, x)
legendre_next(n, x, Pl, Plm1)
legendre_next_m(n, m, x, Pl, Plm1)
```

Arguments

n	Degree of the polynomial
X	Argument of the polynomial
m	Order of the polynomial (for Legendre polynomials of the first kind)
Pl	Value of the Legendre polynomial $(P_l(x))$
Plm1	Value of the Legendre polynomial $(P_{l-1}(x))$

Value

A single numeric value with the computed Legendre polynomial, its derivative, zeros, or related functions.

See Also

Boost Documentation for more details on the mathematical background.

```
# Legendre polynomial of the first kind P_2(0.5) legendre_p(2, 0.5) 
# Derivative of the Legendre polynomial of the first kind P_2'(0.5) legendre_p_prime(2, 0.5)
```

logistic_distribution 49

```
# Zeros of the Legendre polynomial of the first kind P_2
legendre_p_zeros(2)
# Legendre polynomial of the first kind with order 1 P_2^1(0.5)
legendre_p_m(2, 1, 0.5)
# Legendre polynomial of the second kind Q_2(0.5)
legendre_q(2, 0.5)
# Next Legendre polynomial of the first kind P_3(0.5) using P_2(0.5) and P_1(0.5)
legendre_next(2, 0.5, legendre_p(2, 0.5), legendre_p(1, 0.5))
# Next Legendre polynomial of the first kind with order 1 P_3^1(0.5) using P_2^1(0.5) and P_1^1(0.5)
legendre_next_m(2, 1, 0.5, legendre_p_m(2, 1, 0.5), legendre_p_m(1, 1, 0.5))
```

logistic_distribution Logistic Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Logistic distribution.

Usage

```
logistic_pdf(x, location = 0, scale = 1)
logistic_lpdf(x, location = 0, scale = 1)
logistic_cdf(x, location = 0, scale = 1)
logistic_lcdf(x, location = 0, scale = 1)
logistic_quantile(p, location = 0, scale = 1)
```

Arguments

Х	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
р	probability $(0 \le p \le 1)$

anantila

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Examples

```
# Logistic distribution with location = 0, scale = 1
logistic_pdf(0)
logistic_lpdf(0)
logistic_cdf(0)
logistic_lcdf(0)
logistic_quantile(0.5)
```

lognormal_distribution

Log Normal Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Log Normal distribution.

Usage

```
lognormal_pdf(x, location = 0, scale = 1)
lognormal_lpdf(x, location = 0, scale = 1)
lognormal_cdf(x, location = 0, scale = 1)
lognormal_lcdf(x, location = 0, scale = 1)
lognormal_quantile(p, location = 0, scale = 1)
```

Arguments

```
x quantile location location parameter (default is 0) scale scale parameter (default is 1) p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

mapairy_distribution 51

Examples

```
# Log Normal distribution with location = 0, scale = 1
lognormal_pdf(0)
lognormal_lpdf(0)
lognormal_cdf(0)
lognormal_lcdf(0)
lognormal_quantile(0.5)
```

mapairy_distribution Map-Airy Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Map-Airy distribution.

Usage

```
mapairy_pdf(x, location = 0, scale = 1)
mapairy_lpdf(x, location = 0, scale = 1)
mapairy_cdf(x, location = 0, scale = 1)
mapairy_lcdf(x, location = 0, scale = 1)
mapairy_quantile(p, location = 0, scale = 1)
```

Arguments

```
 \begin{array}{ll} x & & \text{quantile} \\ \text{location} & & \text{location parameter (default is 0)} \\ \text{scale} & & \text{scale parameter (default is 1)} \\ \text{p} & & \text{probability } (0 <= p <= 1) \\ \end{array}
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Examples

```
# Distribution only available with Boost version 1.87.0 or later.
## Not run:
# Map-Airy distribution with location 0 and scale 1
mapairy_pdf(3)
mapairy_lpdf(3)
mapairy_cdf(3)
mapairy_lcdf(3)
mapairy_quantile(0.5)
## End(Not run)
```

negative_binomial_distribution

Negative Binomial Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Negative Binomial distribution.

Usage

```
negative_binomial_pdf(x, successes, success_fraction)
negative_binomial_lpdf(x, successes, success_fraction)
negative_binomial_cdf(x, successes, success_fraction)
negative_binomial_lcdf(x, successes, success_fraction)
negative_binomial_quantile(p, successes, success_fraction)
```

Arguments

```
x quantile
successes number of successes (successes >= 0)
success_fraction
probability of success on each trial (0 <= success_fraction <= 1)
p probability (0 <= p <= 1)
```

Value

Boost Documentation for more details on the mathematical background.

Examples

```
negative_binomial_pdf(3, 5, 0.5)
negative_binomial_lpdf(3, 5, 0.5)
negative_binomial_cdf(3, 5, 0.5)
negative_binomial_lcdf(3, 5, 0.5)
negative_binomial_quantile(0.5, 5, 0.5)
```

```
non_central_beta_distribution
```

Noncentral Beta Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Noncentral Beta distribution.

Usage

```
non_central_beta_pdf(x, alpha, beta, lambda)
non_central_beta_lpdf(x, alpha, beta, lambda)
non_central_beta_cdf(x, alpha, beta, lambda)
non_central_beta_lcdf(x, alpha, beta, lambda)
non_central_beta_quantile(p, alpha, beta, lambda)
```

Arguments

```
x quantile (0 \le x \le 1)
alpha first shape parameter (alpha > 0)
beta second shape parameter (beta > 0)
lambda noncentrality parameter (lambda >= 0)
p probability (0 \le p \le 1)
```

Value

Boost Documentation for more details on the mathematical background.

Examples

```
# Noncentral Beta distribution with shape parameters alpha = 2, beta = 3
# and noncentrality parameter lambda = 1
non_central_beta_pdf(0.5, 2, 3, 1)
non_central_beta_lpdf(0.5, 2, 3, 1)
non_central_beta_cdf(0.5, 2, 3, 1)
non_central_beta_lcdf(0.5, 2, 3, 1)
non_central_beta_quantile(0.5, 2, 3, 1)
```

```
{\tt non\_central\_chi\_squared\_distribution}
```

Noncentral Chi-Squared Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Noncentral Chi-Squared distribution.

Usage

```
non_central_chi_squared_pdf(x, df, lambda)
non_central_chi_squared_lpdf(x, df, lambda)
non_central_chi_squared_cdf(x, df, lambda)
non_central_chi_squared_lcdf(x, df, lambda)
non_central_chi_squared_quantile(p, df, lambda)
```

Arguments

```
x quantile df degrees of freedom (df > 0)  
lambda noncentrality parameter (lambda >= 0) p probability (0 <= p <= 1)
```

Value

Boost Documentation for more details on the mathematical background.

Examples

```
## Not run:
# Noncentral Chi-Squared distribution with 3 degrees of freedom and noncentrality
# parameter 1
non_central_chi_squared_pdf(2, 3, 1)
non_central_chi_squared_lpdf(2, 3, 1)
non_central_chi_squared_cdf(2, 3, 1)
non_central_chi_squared_lcdf(2, 3, 1)
non_central_chi_squared_lcdf(2, 3, 1)
## End(Not run)
## End(Not run)
```

```
non_central_t_distribution
```

Noncentral T Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Noncentral T distribution.

Usage

```
non_central_t_pdf(x, df, delta)
non_central_t_lpdf(x, df, delta)
non_central_t_cdf(x, df, delta)
non_central_t_lcdf(x, df, delta)
non_central_t_quantile(p, df, delta)
```

Arguments

```
x quantile

df degrees of freedom (df > 0)

delta noncentrality parameter (delta \ge 0)

p probability (0 <= p <= 1)
```

Value

56 normal_distribution

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Noncentral T distribution with 3 degrees of freedom and noncentrality parameter 1
non_central_t_pdf(0, 3, 1)
non_central_t_lpdf(0, 3, 1)
non_central_t_cdf(0, 3, 1)
non_central_t_lcdf(0, 3, 1)
non_central_t_quantile(0.5, 3, 1)
```

normal_distribution

Normal Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Normal distribution.

Usage

```
normal_pdf(x, mean = 0, sd = 1)
normal_lpdf(x, mean = 0, sd = 1)
normal_cdf(x, mean = 0, sd = 1)
normal_lcdf(x, mean = 0, sd = 1)
normal_quantile(p, mean = 0, sd = 1)
```

Arguments

```
 \begin{array}{lll} x & & \text{quantile} \\ \text{mean} & & \text{mean parameter (default is 0)} \\ \text{sd} & & \text{standard deviation parameter (default is 1)} \\ \text{p} & & \text{probability } (0 <= p <= 1) \\ \end{array}
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

number_series 57

Examples

```
# Normal distribution with mean = 0, sd = 1
normal_pdf(0)
normal_lpdf(0)
normal_cdf(0)
normal_lcdf(0)
normal_quantile(0.5)
```

number_series

Number Series

Description

Functions to compute Bernoulli numbers, tangent numbers, fibonacci numbers, and prime numbers.

Usage

```
bernoulli_b2n(n = NULL, start_index = NULL, number_of_bernoullis_b2n = NULL)
max_bernoulli_b2n()
unchecked_bernoulli_b2n(n)
tangent_t2n(n = NULL, start_index = NULL, number_of_tangent_t2n = NULL)
prime(n)
max_prime()
fibonacci(n)
unchecked_fibonacci(n)
```

Arguments

```
n Index of number to compute (must be a non-negative integer)

start_index The starting index for the range of numbers (must be a non-negative integer)

number_of_bernoullis_b2n

The number of Bernoulli numbers to compute

number_of_tangent_t2n

The number of tangent numbers to compute
```

Details

Efficient computation of Bernoulli numbers, tangent numbers, fibonacci numbers, and prime numbers

The checked_ functions ensure that the input is within valid bounds, while the unchecked_ functions do not perform such checks, allowing for potentially faster computation at the risk of overflow or invalid input.

The range_ functions allow for computing a sequence of numbers starting from a specified index.

The max_ functions return the maximum index for which the respective numbers can be computed using precomputed lookup tables.

Value

A single numeric value for the Bernoulli numbers, tangent numbers, fibonacci numbers, or prime numbers, or a vector of values for ranges.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
bernoulli_b2n(10)
max_bernoulli_b2n()
unchecked_bernoulli_b2n(10)
bernoulli_b2n(start_index = 0, number_of_bernoullis_b2n = 10)
tangent_t2n(10)
tangent_t2n(start_index = 0, number_of_tangent_t2n = 10)
prime(10)
max_prime()
fibonacci(10)
unchecked_fibonacci(10)
```

```
numerical_differentiation
```

Numerical Differentiation

Description

Functions for numerical differentiation using finite difference methods and complex step methods.

Usage

```
finite_difference_derivative(f, x, order = 1)
complex_step_derivative(f, x)
```

numerical_integration 59

Arguments

f	A function to differentiate. It should accept a single numeric value and return a single numeric value.
X	The point at which to evaluate the derivative.
order	The order of accuracy of the derivative to compute. Default is 1.

Value

The approximate value of the derivative at the point x.

Examples

```
# Finite difference derivative of sin(x) at pi/4
finite_difference_derivative(sin, pi / 4)
# Complex step derivative of exp(x) at 1.7
complex_step_derivative(exp, 1.7)
```

numerical_integration Numerical Integration

Description

Functions for numerical integration using various methods such as trapezoidal rule, Gauss-Legendre quadrature, and Gauss-Kronrod quadrature.

Usage

```
trapezoidal(f, a, b, tol = sqrt(.Machine$double.eps), max_refinements = 12)
gauss_legendre(f, a, b, points = 7)
gauss_kronrod(
    f,
        a,
        b,
        points = 15,
        max_depth = 15,
        tol = sqrt(.Machine$double.eps)
)
```

Arguments

- f A function to integrate. It should accept a single numeric value and return a single numeric value.
- a The lower limit of integration.
- b The upper limit of integration.

The tolerance for the approximation. Default is sqrt(.Machine\$double.eps).

max_refinements

The maximum number of refinements to apply. Default is 12.

points

The number of evaluation points to use in the Gauss-Legendre or Gauss-Kronrod quadrature.

max_depth

Sets the maximum number of interval splittings for Gauss-Kronrod permitted before stopping. Set this to zero for non-adaptive quadrature.

Value

A single numeric value with the computed integral.

Examples

```
# Trapezoidal rule integration of sin(x) from 0 to pi
trapezoidal(sin, 0, pi)
# Gauss-Legendre integration of exp(x) from 0 to 1
gauss_legendre(exp, 0, 1, points = 7)
# Adaptive Gauss-Kronrod integration of log(x) from 1 to 2
gauss_kronrod(log, 1, 2, points = 15, max_depth = 10)
```

ooura_fourier_integrals

Ooura Fourier Integrals

Description

Computing Fourier sine and cosine integrals using Ooura's method.

Usage

```
ooura_fourier_sin(
    f,
    omega = 1,
    relative_error_tolerance = sqrt(.Machine$double.eps),
    levels = 8
)

ooura_fourier_cos(
    f,
    omega = 1,
    relative_error_tolerance = sqrt(.Machine$double.eps),
    levels = 8
)
```

owens_t

Arguments

f A function to integrate. It should accept a single numeric value and return a

single numeric value.

omega The frequency parameter for the sine integral.

relative_error_tolerance

The relative error tolerance for the approximation.

levels The number of levels of refinement to apply. Default is 8.

Value

A single numeric value with the computed Fourier sine or cosine integral, with attribute 'relative_error' indicating the relative error of the approximation.

Examples

```
# Fourier sine integral of sin(x) with omega = 1
ooura_fourier_sin(function(x) { 1 / x }, omega = 1)
# Fourier cosine integral of cos(x) with omega = 1
ooura_fourier_cos(function(x) { 1/ (x * x + 1) }, omega = 1)
```

owens_t

Owens T Function

Description

Computes the Owens T function of h and a, giving the probability of the event (X > h and 0 < Y < a * X) where X and Y are independent standard normal random variables.

Usage

```
owens_t(h, a)
```

Arguments

h The first argument of the Owens T function

a The second argument of the Owens T function

Value

The value of the Owens T function at (h, a).

See Also

Boost Documentation for more details on the mathematical background.

```
# Owens T Function
owens_t(1, 0.5)
```

62 pareto_distribution

pareto_distribution Pareto Distri

Pareto Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Pareto distribution.

Usage

```
pareto_pdf(x, shape = 1, scale = 1)

pareto_lpdf(x, shape = 1, scale = 1)

pareto_cdf(x, shape = 1, scale = 1)

pareto_lcdf(x, shape = 1, scale = 1)

pareto_quantile(p, shape = 1, scale = 1)
```

Arguments

```
x quantile
shape shape parameter (default is 1)
scale scale parameter (default is 1)
p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Pareto distribution with shape = 1, scale = 1
pareto_pdf(1)
pareto_lpdf(1)
pareto_cdf(1)
pareto_lcdf(1)
pareto_quantile(0.5)
```

poisson_distribution 63

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Poisson distribution.

Usage

```
poisson_pdf(x, lambda = 1)
poisson_lpdf(x, lambda = 1)
poisson_cdf(x, lambda = 1)
poisson_lcdf(x, lambda = 1)
poisson_quantile(p, lambda = 1)
```

Arguments

```
x quantile
lambda rate parameter (default is 1)
p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

```
# Poisson distribution with lambda = 1
poisson_pdf(0, 1)
poisson_lpdf(0, 1)
poisson_cdf(0, 1)
poisson_lcdf(0, 1)
poisson_quantile(0.5, 1)
```

```
polynomial\_root\_finding \\ \textit{Polynomial Root-Finding}
```

Description

Functions for finding roots of polynomials of various degrees.

Usage

```
quadratic_roots(a, b, c)
cubic_roots(a, b, c, d)
cubic_root_residual(a, b, c, d, root)
cubic_root_condition_number(a, b, c, d, root)
quartic_roots(a, b, c, d, e)
```

Arguments

a	Coefficient of the polynomial term (e.g., for quadratic $ax^2 + bx + c$, a is the coefficient of x^2).
b	Coefficient of the linear term (e.g., for quadratic $ax^2 + bx + c$, b is the coefficient of x).
С	Constant term (e.g., for quadratic $ax^2 + bx + c$, c is the constant).
d	Coefficient of the cubic term (for cubic $ax^3 + bx^2 + cx + d$, d is the constant).
root	The root to evaluate the residual or condition number at.
е	Coefficient of the quartic term (for quartic $ax^4 + bx^3 + cx^2 + dx + e$, e is the constant).

Details

This package provides functions to find roots of quadratic, cubic, and quartic polynomials. The functions return the roots as numeric vectors.

Value

A numeric vector of the polynomial roots, residual, or condition number.

rayleigh_distribution 65

Examples

```
# Example of finding quadratic roots
quadratic_roots(1, -3, 2)
# Example of finding cubic roots
cubic_roots(1, -6, 11, -6)
# Example of finding quartic roots
quartic_roots(1, -10, 35, -50, 24)
# Example of finding cubic root residual
cubic_root_residual(1, -6, 11, -6, 1)
# Example of finding cubic root condition number
cubic_root_condition_number(1, -6, 11, -6, 1)
```

rayleigh_distribution Rayleigh Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Rayleigh distribution.

Usage

```
rayleigh_pdf(x, scale = 1)
rayleigh_lpdf(x, scale = 1)
rayleigh_cdf(x, scale = 1)
rayleigh_lcdf(x, scale = 1)
rayleigh_quantile(p, scale = 1)
```

Arguments

```
x quantile
scale scale parameter (default is 1)
p probability (0 <= p <= 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Examples

```
# Rayleigh distribution with scale = 1
rayleigh_pdf(1)
rayleigh_lpdf(1)
rayleigh_cdf(1)
rayleigh_lcdf(1)
rayleigh_quantile(0.5)
```

rootfinding_and_minimisation

Root-Finding and Minimisation Functions

Description

Functions for root-finding and minimisation using various algorithms.

Usage

```
bisect(
  f,
  lower,
  upper,
  digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
bracket_and_solve_root(
  f,
 guess,
 factor,
 rising,
 digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
toms748_solve(
  f,
  lower,
  upper,
 digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
newton_raphson_iterate(
  f,
  guess,
  lower,
```

```
upper,
 digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
halley_iterate(
 f,
  guess,
 lower,
 upper,
 digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
schroder_iterate(
 f,
 guess,
 lower,
 upper,
 digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
brent_find_minima(
  f,
 lower,
 upper,
 digits = .Machine$double.digits,
 max_iter = .Machine$integer.max
)
```

Arguments

f	A function to find the root of or to minimise. It should take and return a single numeric value for root-finding, or a numeric vector for minimisation.
lower	The lower bound of the interval to search for the root or minimum.
upper	The upper bound of the interval to search for the root or minimum.
digits	The number of significant digits to which the root or minimum should be found. Defaults to double precision.
max_iter	The maximum number of iterations to perform. Defaults to the maximum integer value.
guess	A numeric value that is a guess for the root or minimum.
factor	Size of steps to take when searching for the root.
rising	If TRUE, the function is assumed to be rising, otherwise it is assumed to be falling.

Details

This package provides a set of functions for finding roots of equations and minimising functions using different numerical methods. The methods include bisection, bracket and solve, TOMS 748, Newton-Raphson, Halley's method, Schroder's method, and Brent's method. It also includes functions for finding roots of polynomials (quadratic, cubic, quartic) and computing minima.

Value

A list containing the root or minimum value, the value of the function at that point, and the number of iterations performed.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
f \leftarrow function(x) \ x^2 - 2 bisect(f, lower = 0, upper = 2) bracket\_and\_solve\_root(f, guess = 1, factor = 0.1, rising = TRUE) toms748\_solve(f, lower = 0, upper = 2) f \leftarrow function(x) \ c(x^2 - 2, 2 * x) newton\_raphson\_iterate(f, guess = 1, lower = 0, upper = 2) f \leftarrow function(x) \ c(x^2 - 2, 2 * x, 2) halley\_iterate(f, guess = 1, lower = 0, upper = 2) schroder\_iterate(f, guess = 1, lower = 0, upper = 2) f \leftarrow function(x) \ (x - 2)^2 + 1 brent\_find\_minima(f, lower = 0, upper = 4)
```

saspoint5_distribution

Sas Points Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the $S\alpha S$ Point5 distribution.

Usage

```
saspoint5_pdf(x, location = 0, scale = 1)
saspoint5_lpdf(x, location = 0, scale = 1)
saspoint5_cdf(x, location = 0, scale = 1)
saspoint5_lcdf(x, location = 0, scale = 1)
saspoint5_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability $(0 \le p \le 1)$

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Distribution only available with Boost version 1.87.0 or later.
## Not run:
# SaS Point5 distribution with location 0 and scale 1
saspoint5_pdf(3)
saspoint5_lpdf(3)
saspoint5_cdf(3)
saspoint5_lcdf(3)
saspoint5_lcdf(3)
## End(Not run)
```

```
sinus\_cardinal\_hyperbolic\_functions \\ Sinus\ Cardinal\ and\ Hyperbolic\ Functions
```

Description

Functions to compute the sinus cardinal function and hyperbolic sinus cardinal function.

Usage

```
sinc_pi(x)
sinhc_pi(x)
```

Arguments

x Input value

Value

A single numeric value with the computed sinus cardinal or hyperbolic sinus cardinal function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Sinus cardinal function
sinc_pi(0.5)
# Hyperbolic sinus cardinal function
sinhc_pi(0.5)
```

```
skew_normal_distribution
```

Skew Normal Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Skew Normal distribution.

Usage

```
skew_normal_pdf(x, location = 0, scale = 1, shape = 0)
skew_normal_lpdf(x, location = 0, scale = 1, shape = 0)
skew_normal_cdf(x, location = 0, scale = 1, shape = 0)
skew_normal_lcdf(x, location = 0, scale = 1, shape = 0)
skew_normal_quantile(p, location = 0, scale = 1, shape = 0)
```

Arguments

```
x quantile
location location parameter (default is 0)
scale scale parameter (default is 1)
shape shape parameter (default is 0)
p probability (0 <= p <= 1)
```

Value

spherical_harmonics 71

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Skew Normal distribution with location = 0, scale = 1, shape = 0
skew_normal_pdf(0)
skew_normal_lpdf(0)
skew_normal_cdf(0)
skew_normal_lcdf(0)
skew_normal_quantile(0.5)
```

spherical_harmonics

Spherical Harmonics

Description

Functions to compute spherical harmonics and related functions.

Usage

```
spherical_harmonic(n, m, theta, phi)
spherical_harmonic_r(n, m, theta, phi)
spherical_harmonic_i(n, m, theta, phi)
```

Arguments

n Degree of the spherical harmonic
m Order of the spherical harmonic
theta Polar angle (colatitude)
phi Azimuthal angle (longitude)

Value

A single complex value with the computed spherical harmonic function, or its real and imaginary parts.

See Also

Boost Documentation

72 students_t_distribution

Examples

```
# Spherical harmonic function Y_2^1(0.5, 0.5)
spherical_harmonic(2, 1, 0.5, 0.5)
# Real part of the spherical harmonic function Y_2^1(0.5, 0.5)
spherical_harmonic_r(2, 1, 0.5, 0.5)
# Imaginary part of the spherical harmonic function Y_2^1(0.5, 0.5)
spherical_harmonic_i(2, 1, 0.5, 0.5)
```

```
students\_t\_distribution
```

Student's T Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Student's t distribution.

Usage

```
students_t_pdf(x, df = 1)
students_t_lpdf(x, df = 1)
students_t_cdf(x, df = 1)
students_t_lcdf(x, df = 1)
students_t_quantile(p, df = 1)
```

Arguments

```
x quantile

df degrees of freedom (default is 1)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

triangular_distribution 73

Examples

```
# Student's t distribution with 3 degrees of freedom
students_t_pdf(0, 3)
students_t_lpdf(0, 3)
students_t_cdf(0, 3)
students_t_lcdf(0, 3)
students_t_quantile(0.5, 3)
```

triangular_distribution

Triangular Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Triangular distribution.

Usage

```
triangular_pdf(x, lower = 0, mode = 1, upper = 2)
triangular_lpdf(x, lower = 0, mode = 1, upper = 2)
triangular_cdf(x, lower = 0, mode = 1, upper = 2)
triangular_lcdf(x, lower = 0, mode = 1, upper = 2)
triangular_quantile(p, lower = 0, mode = 1, upper = 2)
```

Arguments

```
x quantile
lower limit of the distribution (default is 0)
mode mode of the distribution (default is 1)
upper upper limit of the distribution (default is 2)
p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

74 uniform_distribution

Examples

```
# Triangular distribution with lower = 0, mode = 1, upper = 2
triangular_pdf(1)
triangular_lpdf(1)
triangular_cdf(1)
triangular_lcdf(1)
triangular_quantile(0.5)
```

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Uniform distribution.

Usage

```
uniform_pdf(x, lower = 0, upper = 1)
uniform_lpdf(x, lower = 0, upper = 1)
uniform_cdf(x, lower = 0, upper = 1)
uniform_lcdf(x, lower = 0, upper = 1)
uniform_quantile(p, lower = 0, upper = 1)
```

Arguments

```
x quantile

lower bound of the distribution (default is 0)

upper upper bound of the distribution (default is 1)

p probability (0 \le p \le 1)
```

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

vector_functionals 75

Examples

```
# Uniform distribution with lower = 0, upper = 1
uniform_pdf(0.5)
uniform_lpdf(0.5)
uniform_cdf(0.5)
uniform_lcdf(0.5)
uniform_quantile(0.5)
```

vector_functionals

Vector Functionals

Description

Functions to compute various vector norms and distances.

Usage

```
10_pseudo_norm(x)
hamming_distance(x, y)
11_norm(x)
11_distance(x, y)
12_norm(x)
12_distance(x, y)
sup_norm(x)
sup_norm(x)
sup_distance(x, y)
1p_norm(x, p)
1p_distance(x, y, p)
total_variation(x)
```

Arguments

x A numeric vector.

y A numeric vector of the same length as x (for distance functions).

A positive integer indicating the order of the norm or distance (for Lp functions).

Value

A single numeric value with the computed norm or distance.

76 weibull_distribution

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# L0 Pseudo Norm
10_pseudo_norm(c(1, 0, 2, 0, 3))
# Hamming Distance
hamming_distance(c(1, 0, 1), c(0, 1, 1))
# L1 Norm
11_{norm(c(1, -2, 3))}
# L1 Distance
11_{\text{distance}}(c(1, -2, 3), c(4, -5, 6))
# L2 Norm
12_{norm}(c(3, 4))
# L2 Distance
12_{\text{distance}}(c(3, 4), c(0, 0))
# Supremum Norm
sup_norm(c(1, -2, 3))
# Supremum Distance
\sup_{0 \le 1} distance(c(1, -2, 3), c(4, -5, 6))
# Lp Norm
lp_norm(c(1, -2, 3), 3)
# Lp Distance
lp_distance(c(1, -2, 3), c(4, -5, 6), 3)
# Total Variation
total\_variation(c(1, 2, 1, 3))
```

weibull_distribution Weibull Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Weibull distribution.

Usage

```
weibull_pdf(x, shape = 1, scale = 1)
weibull_lpdf(x, shape = 1, scale = 1)
weibull_cdf(x, shape = 1, scale = 1)
weibull_lcdf(x, shape = 1, scale = 1)
weibull_quantile(p, shape = 1, scale = 1)
```

zeta 77

Arguments

X	quantile
shape	shape parameter (default is 1)
scale	scale parameter (default is 1)
p	probability $(0 \le p \le 1)$

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Weibull distribution with shape = 1, scale = 1
weibull_pdf(1)
weibull_lpdf(1)
weibull_cdf(1)
weibull_lcdf(1)
weibull_quantile(0.5)
```

zeta

Riemann Zeta Function

Description

Computes the Riemann zeta function $(\zeta(s))$ for argument (z).

Usage

```
zeta(z)
```

Arguments

z Real number input

Value

The value of the Riemann zeta function $(\zeta(z))$.

Examples

```
# Riemann Zeta Function
zeta(2) # Should return pi^2 / 6
```

Index

acosh_boost	beta_distribution, 9
<pre>(inverse_hyperbolic_functions),</pre>	beta_functions, 10
38	<pre>beta_lcdf (beta_distribution), 9</pre>
airy_ai (airy_functions), 3	<pre>beta_lpdf (beta_distribution), 9</pre>
<pre>airy_ai_prime (airy_functions), 3</pre>	beta_pdf (beta_distribution), 9
<pre>airy_ai_zero (airy_functions), 3</pre>	beta_quantile(beta_distribution), 9
airy_bi (airy_functions), 3	betac (beta_functions), 10
<pre>airy_bi_prime (airy_functions), 3</pre>	<pre>binomial_cdf (binomial_distribution), 12</pre>
<pre>airy_bi_zero (airy_functions), 3</pre>	binomial_coefficient
airy_functions, 3	<pre>(factorials_and_binomial_coefficients),</pre>
<pre>arcsine_cdf (arcsine_distribution), 4</pre>	22
arcsine_distribution, 4	binomial_distribution, 12
<pre>arcsine_lcdf (arcsine_distribution), 4</pre>	<pre>binomial_lcdf (binomial_distribution),</pre>
<pre>arcsine_lpdf (arcsine_distribution), 4</pre>	12
<pre>arcsine_pdf (arcsine_distribution), 4</pre>	<pre>binomial_lpdf (binomial_distribution),</pre>
arcsine_quantile	12
<pre>(arcsine_distribution), 4</pre>	<pre>binomial_pdf (binomial_distribution), 12</pre>
asinh_boost	binomial_quantile
<pre>(inverse_hyperbolic_functions),</pre>	<pre>(binomial_distribution), 12</pre>
38	<pre>bisect (rootfinding_and_minimisation),</pre>
atanh_boost	66
<pre>(inverse_hyperbolic_functions),</pre>	bracket_and_solve_root
38	<pre>(rootfinding_and_minimisation),</pre>
	66
basic_functions, 5	brent_find_minima
bernoulli_b2n (number_series), 57	<pre>(rootfinding_and_minimisation),</pre>
bernoulli_cdf (bernoulli_distribution),	66
6	
bernoulli_distribution, 6	<pre>cauchy_cdf (cauchy_distribution), 13</pre>
bernoulli_lcdf	cauchy_distribution, 13
(bernoulli_distribution), 6	<pre>cauchy_lcdf (cauchy_distribution), 13</pre>
bernoulli_lpdf	<pre>cauchy_lpdf (cauchy_distribution), 13</pre>
(bernoulli_distribution), 6	<pre>cauchy_pdf (cauchy_distribution), 13</pre>
<pre>bernoulli_pdf (bernoulli_distribution),</pre>	<pre>cauchy_quantile(cauchy_distribution),</pre>
6	13
bernoulli_quantile	<pre>cbrt (basic_functions), 5</pre>
(bernoulli_distribution), 6	chebyshev_clenshaw_recurrence
bessel_functions, 7	(chebyshev_polynomials), 14
beta_boost (beta_functions), 10	chebyshev_clenshaw_recurrence_ab
beta cdf (beta distribution). 9	(chebyshev polynomials), 14

<pre>chebyshev_next (chebyshev_polynomials),</pre>	ellint_d(elliptic_integrals), 17
14	ellint_rc(elliptic_integrals), 17
chebyshev_polynomials, 14	ellint_rd(elliptic_integrals), 17
<pre>chebyshev_t (chebyshev_polynomials), 14</pre>	ellint_rf(elliptic_integrals), 17
chebyshev_t_prime	ellint_rg(elliptic_integrals), 17
<pre>(chebyshev_polynomials), 14</pre>	ellint_rj(elliptic_integrals), 17
<pre>chebyshev_u (chebyshev_polynomials), 14</pre>	elliptic_integrals, 17
chi_squared_cdf	erf (error_functions), 19
<pre>(chi_squared_distribution), 15</pre>	<pre>erf_inv (error_functions), 19</pre>
<pre>chi_squared_distribution, 15</pre>	erfc(error_functions), 19
chi_squared_lcdf	erfc_inv (error_functions), 19
<pre>(chi_squared_distribution), 15</pre>	error_functions, 19
chi_squared_lpdf	exp_sinh
<pre>(chi_squared_distribution), 15</pre>	<pre>(double_exponential_quadrature),</pre>
chi_squared_pdf	16
(chi_squared_distribution), 15	<pre>expint_ei (exponential_integrals), 21</pre>
chi_squared_quantile	<pre>expint_en (exponential_integrals), 21</pre>
(chi_squared_distribution), 15	<pre>expm1_boost(basic_functions), 5</pre>
complex_step_derivative	exponential_cdf
(numerical_differentiation), 58	(exponential_distribution), 20
cos_pi (basic_functions), 5	exponential_distribution, 20
cubic_root_condition_number	exponential_integrals, 21
<pre>(polynomial_root_finding), 64</pre>	exponential_lcdf
cubic_root_residual	(exponential_distribution), 20
<pre>(polynomial_root_finding), 64</pre>	exponential_lpdf
<pre>cubic_roots (polynomial_root_finding),</pre>	(exponential_distribution), 20
64	exponential_pdf
<pre>cyl_bessel_i (bessel_functions), 7</pre>	$(exponential_distribution), 20$
<pre>cyl_bessel_i_prime (bessel_functions), 7</pre>	exponential_quantile
<pre>cyl_bessel_j (bessel_functions), 7</pre>	$(exponential_distribution), 20$
<pre>cyl_bessel_j_prime (bessel_functions), 7</pre>	extreme_value_cdf
<pre>cyl_bessel_j_zero (bessel_functions), 7</pre>	<pre>(extreme_value_distribution),</pre>
<pre>cyl_bessel_k (bessel_functions), 7</pre>	21
<pre>cyl_bessel_k_prime (bessel_functions), 7</pre>	<pre>extreme_value_distribution, 21</pre>
<pre>cyl_hankel_1 (hankel_functions), 30</pre>	extreme_value_lcdf
cyl_hankel_2 (hankel_functions), 30	<pre>(extreme_value_distribution),</pre>
<pre>cyl_neumann (bessel_functions), 7</pre>	21
<pre>cyl_neumann_prime (bessel_functions), 7</pre>	extreme_value_lpdf
<pre>cyl_neumann_zero (bessel_functions), 7</pre>	<pre>(extreme_value_distribution),</pre>
	21
digamma_boost (gamma_functions), 26	extreme_value_pdf
double_exponential_quadrature, 16	<pre>(extreme_value_distribution),</pre>
double_factorial	21
<pre>(factorials_and_binomial_coefficie</pre>	ntsextreme_value_quantile
22	<pre>(extreme_value_distribution),</pre>
	21
ellint_1 (elliptic_integrals), 17	
ellint_2 (elliptic_integrals), 17	factorial_boost
ellint_3 (elliptic_integrals), 17	<pre>(factorials_and_binomial_coefficients)</pre>

22	geometric_lpdf
factorials_and_binomial_coefficients,	(geometric_distribution), 29
22	<pre>geometric_pdf (geometric_distribution),</pre>
falling_factorial	29
(factorials_and_binomial_coefficients)	ntsgeometric_quantile
22	(geometric_distribution), 29
fibonacci (number_series), 57	
finite_difference_derivative	halley_iterate
(numerical_differentiation), 58	(rootfinding_and_minimisation),
fisher_f_cdf (fisher_f_distribution), 24	66
fisher_f_distribution, 24	hamming_distance(vector_functionals),
<pre>fisher_f_lcdf (fisher_f_distribution),</pre>	75
24	hankel_functions, 30
<pre>fisher_f_lpdf (fisher_f_distribution),</pre>	hermite (hermite_polynomials), 30
24	hermite_next (hermite_polynomials), 30
<pre>fisher_f_pdf (fisher_f_distribution), 24</pre>	hermite_polynomials, 30
fisher_f_quantile	heuman_lambda (elliptic_integrals), 17
(fisher_f_distribution), 24	holtsmark_cdf (holtsmark_distribution),
, – – ,,	31
gamma_cdf (gamma_distribution), 25	holtsmark_distribution, 31
gamma_distribution, 25	holtsmark_lcdf
gamma_functions, 26	
gamma_lcdf (gamma_distribution), 25	(holtsmark_distribution), 31
gamma_lpdf (gamma_distribution), 25	holtsmark_lpdf
gamma_p (gamma_functions), 26	(holtsmark_distribution), 31
gamma_p_derivative(gamma_functions), 26	holtsmark_pdf (holtsmark_distribution),
gamma_p_inv (gamma_functions), 26	31
gamma_p_inva (gamma_functions), 26	holtsmark_quantile
	(holtsmark_distribution), 31
gamma_pdf (gamma_distribution), 25	hyperexponential_cdf
gamma_q (gamma_functions), 26	<pre>(hyperexponential_distribution),</pre>
gamma_q_inv (gamma_functions), 26	32
gamma_q_inva (gamma_functions), 26	hyperexponential_distribution, 32
gamma_quantile (gamma_distribution), 25	hyperexponential_lcdf
gauss_kronrod (numerical_integration),	<pre>(hyperexponential_distribution),</pre>
59	32
<pre>gauss_legendre (numerical_integration),</pre>	hyperexponential_lpdf
59	<pre>(hyperexponential_distribution),</pre>
gegenbauer (gegenbauer_polynomials), 28	32
gegenbauer_derivative	hyperexponential_pdf
(gegenbauer_polynomials), 28	(hyperexponential_distribution),
gegenbauer_polynomials, 28	32
gegenbauer_prime	hyperexponential_quantile
(gegenbauer_polynomials), 28	<pre>(hyperexponential_distribution),</pre>
<pre>geometric_cdf (geometric_distribution),</pre>	32
29	hypergeometric_0F1
geometric_distribution, 29	(hypergeometric_functions), 34
geometric_lcdf	hypergeometric_1F0
(geometric_distribution), 29	<pre>(hypergeometric_functions), 34</pre>

nypergeometric_IFI	inverse_cni_squared_quantile
(hypergeometric_functions), 34	<pre>(inverse_chi_squared_distribution)</pre>
hypergeometric_1F1_regularized	35
<pre>(hypergeometric_functions), 34</pre>	inverse_gamma_cdf
hypergeometric_2F0	<pre>(inverse_gamma_distribution),</pre>
<pre>(hypergeometric_functions), 34</pre>	36
hypergeometric_cdf	inverse_gamma_distribution, 36
(hypergeometric_distribution),	inverse_gamma_lcdf
33	(inverse_gamma_distribution),
hypergeometric_distribution, 33	36
hypergeometric_functions, 34	inverse_gamma_lpdf
hypergeometric_lcdf	(inverse_gamma_distribution),
<pre>(hypergeometric_distribution),</pre>	36
33	inverse_gamma_pdf
hypergeometric_lpdf	<pre>(inverse_gamma_distribution),</pre>
<pre>(hypergeometric_distribution),</pre>	36
33	inverse_gamma_quantile
hypergeometric_pdf	(inverse_gamma_distribution),
<pre>(hypergeometric_distribution),</pre>	36
33	inverse_gaussian_cdf
hypergeometric_pFq	(inverse_gaussian_distribution),
(hypergeometric_functions), 34	37
hypergeometric_quantile	
(hypergeometric_distribution),	inverse_gaussian_distribution, 37
33	inverse_gaussian_lcdf
hypot (basic_functions), 5	<pre>(inverse_gaussian_distribution),</pre>
Typot (basic_runctions), 3	37
ibeta(beta_functions), 10	inverse_gaussian_lpdf
ibeta_derivative (beta_functions), 10	(inverse_gaussian_distribution),
	37
ibeta_inv (beta_functions), 10	inverse_gaussian_pdf
ibeta_inva (beta_functions), 10	(inverse_gaussian_distribution),
ibeta_invb (beta_functions), 10	37
ibetac (beta_functions), 10	inverse_gaussian_quantile
ibetac_inv (beta_functions), 10	(inverse_gaussian_distribution),
ibetac_inva (beta_functions), 10	37
ibetac_invb (beta_functions), 10	inverse_hyperbolic_functions, 38
inverse_chi_squared_cdf	Thver se_hyperbolic_runctions, 36
<pre>(inverse_chi_squared_distribution),</pre>	
35	jacobi(jacobi_polynomials),41
<pre>inverse_chi_squared_distribution, 35</pre>	<pre>jacobi_cd (jacobi_elliptic_functions),</pre>
inverse_chi_squared_lcdf	39
<pre>(inverse_chi_squared_distribution),</pre>	<pre>jacobi_cn(jacobi_elliptic_functions),</pre>
35	39
inverse_chi_squared_lpdf	<pre>jacobi_cs (jacobi_elliptic_functions),</pre>
(inverse_chi_squared_distribution),	39
35	<pre>jacobi_dc (jacobi_elliptic_functions),</pre>
inverse_chi_squared_pdf	39
	
<pre>(inverse_chi_squared_distribution),</pre>	<pre>jacobi_derivative(jacobi_polynomials),</pre>
35	41

<pre>jacobi_dn(jacobi_elliptic_functions),</pre>	<pre>jacobi_zeta(elliptic_integrals), 17</pre>
jacobi_double_prime	kolmogorov_smirnov_cdf
<pre>(jacobi_polynomials), 41</pre>	<pre>(kolmogorov_smirnov_distribution).</pre>
<pre>jacobi_ds (jacobi_elliptic_functions),</pre>	43
39	kolmogorov_smirnov_distribution, 43
jacobi_elliptic	kolmogorov_smirnov_lcdf
(jacobi_elliptic_functions), 39	<pre>(kolmogorov_smirnov_distribution).</pre>
jacobi_elliptic_functions, 39	43
<pre>jacobi_nc (jacobi_elliptic_functions),</pre>	kolmogorov_smirnov_lpdf
39	<pre>(kolmogorov_smirnov_distribution),</pre>
<pre>jacobi_nd (jacobi_elliptic_functions),</pre>	43
39	kolmogorov_smirnov_pdf
<pre>jacobi_ns (jacobi_elliptic_functions),</pre>	<pre>(kolmogorov_smirnov_distribution)</pre>
39	43
jacobi_polynomials, 41	kolmogorov_smirnov_quantile
jacobi_prime (jacobi_polynomials), 41	<pre>(kolmogorov_smirnov_distribution);</pre>
jacobi_sc(jacobi_elliptic_functions),	43
39	
 -	<pre>10_pseudo_norm (vector_functionals), 75</pre>
<pre>jacobi_sd(jacobi_elliptic_functions),</pre>	<pre>11_distance (vector_functionals), 75</pre>
39	<pre>11_norm(vector_functionals), 75</pre>
<pre>jacobi_sn(jacobi_elliptic_functions),</pre>	12_distance (vector_functionals), 75
39	<pre>12_norm(vector_functionals), 75</pre>
<pre>jacobi_theta1 (jacobi_theta_functions),</pre>	laguerre (laguerre_polynomials), 44
42	<pre>laguerre_m (laguerre_polynomials), 44</pre>
jacobi_theta1tau	<pre>laguerre_next (laguerre_polynomials), 44</pre>
(jacobi_theta_functions), 42	<pre>laguerre_next_m(laguerre_polynomials),</pre>
<pre>jacobi_theta2(jacobi_theta_functions),</pre>	44
42	laguerre_polynomials,44
jacobi_theta2tau	lambert_w0 (lambert_w_function), 45
(jacobi_theta_functions), 42	<pre>lambert_w0_prime (lambert_w_function),</pre>
<pre>jacobi_theta3 (jacobi_theta_functions),</pre>	45
42	lambert_w_function, 45
jacobi_theta3m1	<pre>lambert_wm1 (lambert_w_function), 45</pre>
<pre>(jacobi_theta_functions), 42</pre>	<pre>lambert_wm1_prime (lambert_w_function),</pre>
jacobi_theta3m1tau	45
<pre>(jacobi_theta_functions), 42</pre>	<pre>landau_cdf (landau_distribution), 46</pre>
jacobi_theta3tau	landau_distribution, 46
<pre>(jacobi_theta_functions), 42</pre>	landau_lcdf (landau_distribution), 46
<pre>jacobi_theta4(jacobi_theta_functions),</pre>	landau_lpdf (landau_distribution), 46
42	landau_pdf (landau_distribution), 46
jacobi_theta4m1	landau_quantile (landau_distribution),
(jacobi_theta_functions), 42	46
jacobi_theta4m1tau	laplace_cdf (laplace_distribution), 47
(jacobi_theta_functions), 42	laplace_distribution, 47
jacobi_theta4tau	laplace_lcdf (laplace_distribution), 47
(jacobi_theta_functions), 42	laplace_lpdf (laplace_distribution), 47
jacobi_theta_functions, 42	laplace_pdf (laplace_distribution), 47

laplace_quantile	max_factorial
(laplace_distribution), 47	<pre>(factorials_and_binomial_coefficients),</pre>
<pre>legendre_next (legendre_polynomials), 48</pre>	22
<pre>legendre_next_m (legendre_polynomials),</pre>	<pre>max_prime (number_series), 57</pre>
<pre>legendre_p (legendre_polynomials), 48</pre>	negative_binomial_cdf
<pre>legendre_p_m (legendre_polynomials), 48</pre>	<pre>(negative_binomial_distribution),</pre>
legendre_p_prime	52
(legendre_polynomials), 48	${\sf negative_binomial_distribution}, 52$
legendre_p_zeros	negative_binomial_lcdf
(legendre_polynomials), 48	<pre>(negative_binomial_distribution),</pre>
legendre_polynomials, 48	52
<pre>legendre_q (legendre_polynomials), 48</pre>	negative_binomial_lpdf
lgamma_boost (gamma_functions), 26	<pre>(negative_binomial_distribution),</pre>
<pre>log1p_boost (basic_functions), 5</pre>	52
log_hypergeometric_1F1	negative_binomial_pdf
(hypergeometric_functions), 34	<pre>(negative_binomial_distribution),</pre>
logistic_cdf (logistic_distribution), 49	52
logistic_distribution, 49	negative_binomial_quantile
logistic_lcdf (logistic_distribution),	<pre>(negative_binomial_distribution),</pre>
49	52
<pre>logistic_lpdf (logistic_distribution),</pre>	newton_raphson_iterate
49	<pre>(rootfinding_and_minimisation),</pre>
<pre>logistic_pdf (logistic_distribution), 49</pre>	66
logistic_quantile	non_central_beta_cdf
(logistic_distribution), 49	(non_central_beta_distribution),
<pre>lognormal_cdf (lognormal_distribution),</pre>	53
50	non_central_beta_distribution, 53
lognormal_distribution, 50	non_central_beta_lcdf
lognormal_lcdf	<pre>(non_central_beta_distribution), 53</pre>
(lognormal_distribution), 50	
lognormal_lpdf	non_central_beta_lpdf
(lognormal_distribution), 50	<pre>(non_central_beta_distribution), 53</pre>
<pre>lognormal_pdf (lognormal_distribution),</pre>	non_central_beta_pdf
50	(non_central_beta_distribution),
<pre>lognormal_quantile</pre>	53
(lognormal_distribution), 50	non_central_beta_quantile
<pre>lp_distance (vector_functionals), 75</pre>	(non_central_beta_distribution),
<pre>lp_norm(vector_functionals), 75</pre>	53
	non_central_chi_squared_cdf
<pre>mapairy_cdf (mapairy_distribution), 51</pre>	(non_central_chi_squared_distribution),
mapairy_distribution, 51	54
mapairy_lcdf (mapairy_distribution), 51	non_central_chi_squared_distribution,
mapairy_lpdf (mapairy_distribution), 51	54
mapairy_pdf (mapairy_distribution), 51	non_central_chi_squared_lcdf
mapairy_quantile	(non_central_chi_squared_distribution),
(mapairy_distribution), 51	54
max_bernoulli_b2n (number_series), 57	non_central_chi_squared_lpdf

<pre>(non_central_chi_squared_distribution</pre>	poisson_cdf (poisson_distribution), 63
54	poisson_distribution, 63
non_central_chi_squared_pdf	<pre>poisson_lcdf (poisson_distribution), 63</pre>
<pre>(non_central_chi_squared_distribution</pre>	poisson_lpdf(poisson_distribution), 63
54	<pre>poisson_pdf (poisson_distribution), 63</pre>
non_central_chi_squared_quantile	poisson_quantile
<pre>(non_central_chi_squared_distribution</pre>	(poisson_distribution), 63
54	polygamma (gamma_functions), 26
non_central_t_cdf	polynomial_root_finding, 64
<pre>(non_central_t_distribution),</pre>	powm1 (basic_functions), 5
55	prime (number_series), 57
non_central_t_distribution, 55	
non_central_t_lcdf	quadratic_roots
<pre>(non_central_t_distribution),</pre>	(polynomial_root_finding), 64
55	quartic_roots
non_central_t_lpdf	(polynomial_root_finding), 64
(non_central_t_distribution),	rayleigh_cdf (rayleigh_distribution), 65
55	rayleigh_distribution, 65
non_central_t_pdf	rayleigh_lcdf (rayleigh_distribution),
(non_central_t_distribution),	65
55	rayleigh_lpdf (rayleigh_distribution),
non_central_t_quantile	65
(non_central_t_distribution),	rayleigh_pdf (rayleigh_distribution), 65
55	rayleigh_quantile
normal_cdf (normal_distribution), 56	(rayleigh_distribution), 65
normal_distribution, 56	rising_factorial
normal_lcdf (normal_distribution), 56	(factorials_and_binomial_coefficients),
normal_lpdf (normal_distribution), 56	22
normal_pdf (normal_distribution), 56	rootfinding_and_minimisation, 66
normal_quantile (normal_distribution),	rsqrt (basic_functions), 5
56	rsqrt (basic_runctions), 3
number_series, 57	<pre>saspoint5_cdf (saspoint5_distribution),</pre>
numerical_differentiation, 58	68
numerical_integration, 59	saspoint5_distribution, 68
Trainer 1841_Integracion, 57	saspoint5_lcdf
ooura_fourier_cos	(saspoint5_distribution), 68
(ooura_fourier_integrals), 60	saspoint5_lpdf
ooura_fourier_integrals, 60	(saspoint5_distribution), 68
ooura_fourier_sin	<pre>saspoint5_pdf (saspoint5_distribution),</pre>
(ooura_fourier_integrals), 60	68
owens_t, 61	saspoint5_quantile
	(saspoint5_distribution), 68
pareto_cdf (pareto_distribution), 62	schroder_iterate
pareto_distribution, 62	(rootfinding_and_minimisation),
pareto_lcdf (pareto_distribution), 62	66
pareto_lpdf (pareto_distribution), 62	sin_pi (basic_functions), 5
pareto_pdf (pareto_distribution), 62	sinc_pi
pareto_quantile (pareto_distribution),	<pre>(sinus_cardinal_hyperbolic_functions),</pre>
62	69

sinh_sinh	tanh_sinh
<pre>(double_exponential_quadrature),</pre>	<pre>(double_exponential_quadrature),</pre>
16	16
sinhc_pi	tgamma (gamma_functions), 26
(sinus_cardinal_hyperbolic_functions),tgamma1pm1 (gamma_functions), 26
69	tgamma_delta_ratio(gamma_functions), 26
<pre>sinus_cardinal_hyperbolic_functions,</pre>	tgamma_lower (gamma_functions), 26
69	tgamma_ratio (gamma_functions), 26
skew_normal_cdf	tgamma_upper (gamma_functions), 26
(skew_normal_distribution), 70	toms748_solve
skew_normal_distribution, 70	<pre>(rootfinding_and_minimisation),</pre>
skew_normal_lcdf	66
(skew_normal_distribution), 70	total_variation(vector_functionals), 75
skew_normal_lpdf	trapezoidal (numerical_integration), 59
(skew_normal_distribution), 70	triangular_cdf
skew_normal_pdf	(triangular_distribution), 73
(skew_normal_distribution), 70	triangular_distribution, 73
skew_normal_quantile	triangular_lcdf
(skew_normal_distribution), 70	(triangular_distribution), 73
<pre>sph_bessel (bessel_functions), 7</pre>	triangular_lpdf
<pre>sph_bessel_prime (bessel_functions), 7</pre>	(triangular_distribution), 73
sph_hankel_1 (hankel_functions), 30	triangular_pdf
sph_hankel_2 (hankel_functions), 30	(triangular_distribution), 73
sph_neumann (bessel_functions), 7	triangular_quantile
sph_neumann_prime (bessel_functions), 7	(triangular_distribution), 73
spherical_harmonic	trigamma_boost (gamma_functions), 26
(spherical_harmonics), 71	
spherical_harmonic_i	unchecked_bernoulli_b2n
(spherical_harmonics), 71	(number_series), 57
spherical_harmonic_r	unchecked_factorial
(spherical_harmonics), 71	(factorials_and_binomial_coefficients),
spherical_harmonics, 71	22
sqrt1pm1 (basic_functions), 5	unchecked_fibonacci (number_series), 57
students_t_cdf	uniform_cdf (uniform_distribution), 74
(students_t_distribution), 72	uniform_distribution, 74
students_t_distribution, 72	uniform_lcdf (uniform_distribution), 74
students_t_lcdf	uniform_lpdf (uniform_distribution), 74
(students_t_distribution), 72	uniform_pdf (uniform_distribution), 74
students_t_lpdf	uniform_quantile
(students_t_distribution), 72	(uniform_distribution), 74
students_t_pdf	(dilitoriii_distribution), /4
(students_t_distribution), 72	constant Constitution 1. 75
students_t_quantile	vector_functionals, 75
·	
(students_t_distribution), 72	weibull_cdf (weibull_distribution), 76
sup_distance (vector_functionals), 75	weibull_distribution, 76
<pre>sup_norm(vector_functionals), 75</pre>	weibull_lcdf (weibull_distribution), 76
	weibull_lpdf (weibull_distribution), 76
tangent_t2n (number_series), 57	<pre>weibull_pdf (weibull_distribution), 76</pre>