Package 'ADtools'

November 9, 2020

Type Package

Title Automatic Differentiation Toolbox

Version 0.5.4

Maintainer Chun Fung Kwok <kwokcf@unimelb.edu.au>

Description Implements the forward-mode automatic differentiation for multivariate functions using the matrix-calculus notation from Magnus and Neudecker (2019) <doi:10.1002/9781119541219>. Two key features of the package are: (i) it incorporates various optimisation strategies to improve performance; this includes applying memoisation to cut down object construction time, using sparse matrix representation to speed up derivative calculation, and creating specialised matrix operations to reduce computation time; (ii) it supports differentiating random variates with respect to their parameters, targeting Markov chain Monte Carlo (MCMC) and general simulation-based applications.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests testthat, covr, knitr, rmarkdown, pryr, MCMCpack

Depends R (>= 3.6.0), methods, Matrix

Imports purrr, dplyr, magrittr, assertthat, mvtnorm, Rcpp

LinkingTo Rcpp, RcppArmadillo

Collate 'ADtools.R' 'RcppExports.R' 'auto_diff.R' 'class_dual_construction.R' 'class_dual_def.R' 'dual_list_operations.R' 'finite_diff.R' 'optim_special_matrix.R' 'interface_special_matrix.R' 'mc_arithmetic_1_cross_class.R' 'mc_arithmetic_2_same_class.R' 'mc_arithmetic_3_d_arithmetic.R' 'mc_arithmetic_4_d_arithmetic_primitives.R' 'mc_generic.R' 'mc_math_elementary.R' 'mc_math_sum.R' 'mc_matrix_ops.R' 'mc_rv_density.R' 'mc_rv_sim_normal.R' 'mc_rv_sim_student_t.R' 'mc_rv_simulation.R' 'mc_structure_binding.R' 'mc_structure_reshape.R' 'mc_structure_subsetting.R'

'optim_kronecker_product.R'
'optim_matrix_chain_multiplication.R' 'utils.R'

VignetteBuilder knitr

URL https://github.com/kcf-jackson/ADtools

BugReports https://github.com/kcf-jackson/ADtools/issues

NeedsCompilation yes

Author Chun Fung Kwok [aut, cre] (<https://orcid.org/0000-0002-0716-3879>),
Dan Zhu [aut] (<https://orcid.org/0000-0003-1487-2232>),
Liana Jacobi [aut] (<https://orcid.org/0000-0001-7210-0500>)

Repository CRAN

Date/Publication 2020-11-09 10:10:02 UTC

R topics documented:

`*,ANY,dual-method` *(Element-wise) Multiplication of 'dual'-class objects (ANY-dual)*

Description

(Element-wise) Multiplication of 'dual'-class objects (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
e1 * e2
```

Arguments

`e1`	"ANY" object.
`e2`	A "dual" object.

`*,dual,ANY-method` *(Element-wise) Multiplication of 'dual'-class objects (dual-ANY)*

Description

(Element-wise) Multiplication of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
e1 * e2
```

Arguments

`e1`	A "dual" object.
`e2`	"ANY" object.

`*,dual,dual-method` *(Element-wise) Multiplication of 'dual'-class objects (dual-dual)*

Description

(Element-wise) Multiplication of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
e1 * e2
```

Arguments

`e1`	A "dual" object.
`e2`	A "dual" object.

`+,ANY,dual-method` *Addition of 'dual'-class objects (ANY-dual)*

Description

Addition of 'dual'-class objects (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
e1 + e2
```

Arguments

`e1`	"ANY" object.
`e2`	A "dual" object.

+,dual,ANY-method *Addition of 'dual'-class objects (dual-ANY)*

Description

Addition of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
e1 + e2
```

Arguments

e1	A "dual" object.
e2	"ANY" object.

+,dual,dual-method *Addition of 'dual'-class objects (dual-dual)*

Description

Addition of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
e1 + e2
```

Arguments

e1	A "dual" object.
e2	A "dual" object.

-,ANY,dual-method *Subtraction of 'dual'-class objects (dual-ANY)*

Description

Subtraction of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'ANY,dual'
e1 - e2
```

Arguments

e1	"ANY" object.
e2	A "dual" object.

-,dual,ANY-method *Subtraction of 'dual'-class objects (dual-ANY)*

Description

Subtraction of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
e1 - e2
```

Arguments

e1	A "dual" object.
e2	"ANY" object.

-,dual,dual-method *Subtraction of 'dual'-class objects (dual-dual)*

Description

Subtraction of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
e1 - e2
```

Arguments

e1	A "dual" object.
e2	A "dual" object.

-,dual,missing-method *Subtraction of 'dual'-class objects (unary dual)*

Description

Subtraction of 'dual'-class objects (unary dual)

Usage

```
## S4 method for signature 'dual,missing'
e1 - e2
```

Arguments

e1	A "dual" object.
e2	"missing" object.

/,ANY,dual-method *(Element-wise) Division of 'dual'-class objects (ANY-dual)*

Description

(Element-wise) Division of 'dual'-class objects (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
e1 / e2
```

Arguments

e1	"ANY" object.
e2	A "dual" object.

/,dual,ANY-method *(Element-wise) Division of 'dual'-class objects (dual-ANY)*

Description

(Element-wise) Division of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
e1 / e2
```

Arguments

e1	A "dual" object.
e2	"ANY" object.

/,dual,dual-method *(Element-wise) Division of 'dual'-class objects (dual-dual)*

Description

(Element-wise) Division of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
e1 / e2
```

Arguments

e1	A "dual" object.
e2	A "dual" object.

ABxI *Compute A (B %x% I)*

Description

Compute A (B %x% I)

Usage

```
ABxI(A, B)
```

Arguments

A	numeric matrix.
B	numeric matrix.

`add_vector_to_matrix_column`
Add a column vector to each column of a matrix

Description

Add a column vector to each column of a matrix

Usage

```
add_vector_to_matrix_column(v0, m0)
```

Arguments

`v0`	Column vector
`m0`	Matrix

`add_vector_to_matrix_row`
Add a column vector to each row of a matrix

Description

Add a column vector to each row of a matrix

Usage

```
add_vector_to_matrix_row(v0, m0)
```

Arguments

`v0`	Column vector
`m0`	Matrix

AIxC *Compute A (I %x% C)*

Description

Compute A (I %x% C)

Usage

```
AIxC(A, C)
```

Arguments

A	numeric matrix.
C	numeric matrix.

as.matrix.dual *Coerce the first component of the dual object into a matrix.*

Description

Coerce the first component of the dual object into a matrix.

Usage

```
## S3 method for class 'dual'
as.matrix(x, ...)

## S4 method for signature 'dual'
as.matrix(x, ...)
```

Arguments

x	A "dual" object.
...	Unused.

as.vector,dual-method *Coerce the first component of the dual object into a vector.*

Description

Coerce the first component of the dual object into a vector.

Usage

```
## S4 method for signature 'dual'
as.vector(x)
```

Arguments

x	A "dual" object.

auto_diff *Automatic differentiation*

Description

Automatic differentiation

Usage

```
auto_diff(f, wrt = NULL, at)
```

Arguments

f	A function of which the derivative is sought.
wrt	A character vector; the name of the variables to differentiate with respect to.
at	A named list of variables; the point at which the derivative is evaluated.

Value

A dual number with components "x" and "dx". The first gives the value of ‘f‘, and the second gives the derivative of ‘f‘.

Examples

```
f <- function(y, X, beta) { y - X %*% beta }
auto_diff(
  f, wrt = "beta",
  at = list(beta = c(5,6), X = matrix(1:4, 2, 2), y = c(2,3))
)

g <- function(X, Y) { X %*% Y }
X <- randn(2, 2)
Y <- randn(2, 2)
auto_diff(g, at = list(X = X, Y = Y))
```

---

band_matrix *Band matrix*

Description

Band matrix

Usage

```
band_matrix(nr, nc, x = 1)
```

Arguments

nr	integer; row dimension.
nc	integer; column dimension.
x	A scalar or a vector to be placed on the diagonal of the matrix.

Examples

```
band_matrix(2, 3)
band_matrix(2, 3, c(999))
band_matrix(2, 3, c(999, 111))
```

band_matrix0 *Band matrix (memoised)*

Description

Band matrix (memoised)

Usage

```
band_matrix0(...)
```

Arguments

...	'nr' integer; number of rows. 'nc' integer; number of columns. 'x' A scalar or a vector to be placed on the diagonal of the matrix. Use empty argument to clear the cache.

BxAZ *Post-multiplying a kronecker product*

Description

Compute (B %x% A) Z

Usage

```
BxAZ(B, A, Z)
```

Arguments

B	numeric matrix.
A	numeric matrix.
Z	numeric matrix.

BxID *Compute (B %x% I) D*

Description

Compute (B %x% I) D

Usage

```
BxID(B, D)
```

Arguments

B	numeric matrix.
D	numeric matrix.

cbind2,ANY,dual-method *Combine 'dual'-class objects by Columns (ANY-dual)*

Description

Combine 'dual'-class objects by Columns (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
cbind2(x, y)
```

Arguments

x	ANY object.
y	A "dual" object.

cbind2,dual,ANY-method

Combine 'dual'-class objects by Columns (dual-ANY)

Description

Combine 'dual'-class objects by Columns (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
cbind2(x, y)
```

Arguments

x	A "dual" object.
y	ANY object.

cbind2,dual,dual-method

Combine 'dual'-class objects by Columns (dual-dual)

Description

Combine 'dual'-class objects by Columns (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
cbind2(x, y)
```

Arguments

x	A "dual" object.
y	A "dual" object.

chol,dual-method *Cholesky decomposition of 'dual'-class objects*

Description

Cholesky decomposition of 'dual'-class objects

Usage

```
## S4 method for signature 'dual'
chol(x)
```

Arguments

x	A "dual" object.

Note

The Cholesky decomposition used in this function returns an upper triangular matrix.

chol0 *Cholesky decomposition*

Description

Cholesky decomposition

Usage

```
chol0(x)
```

Arguments

x	A numeric matrix.

Details

`chol0` is implemented as (t o chol o t) becauase the Cholesky decomposition in R (`chol`)

1. returns an upper-triangle matrix;
2. uses only the upper half of the matrix when the matrix is real.

This does not match with the usual math notation of A = LL^T, where L is a lower triangular matrix. (Note that the Cholesky-Banachiewicz algorithm for example only uses the lower triangular part of A to compute L, so one should expect d L / d A to look like something you get by differentiating a lower triangular matrix w.r.t. a lower triangular matrix, i.e. the resulting Jacobian matrix is also lower triangular.)

To convert the R version of chol to the usual math version of chol, we define `chol0 <-function(x) { t(chol(t(x))) }` The first transpose ensures the output is lower-triangular, and the second ensures the lower-triangular part of the input is used. Now, chol0

1. returns a lower-trianglar matrix
2. uses only the lower half of the matrix when the matrix is real

and this is what we want. (Additional note: finite-differencing with Cholesky is not too accurate due to the many floating point operations involved.)

Note

This function uses only the lower-triangular part of the input and returns a lower-triangular matrix.

`chol0,dual-method` *Cholesky decomposition of 'dual'-class objects*

Description

Cholesky decomposition of 'dual'-class objects

Usage

```
## S4 method for signature 'dual'
chol0(x)
```

Arguments

`x`	A "dual" object.

Note

The Cholesky decomposition used in this function returns a lower triangular matrix.

colMeans,dual-method *Column mean of a matrix.*

Description

Column mean of a matrix.

Usage

```
## S4 method for signature 'dual'
colMeans(x)
```

Arguments

x	A "dual" object.

colSums,dual-method *Column sum of a matrix.*

Description

Column sum of a matrix.

Usage

```
## S4 method for signature 'dual'
colSums(x)
```

Arguments

x	A "dual" object.

commutation_matrix *Commutation matrix*

Description

Suppose A is a (r x c) matrix, then a (rc x rc) matrix K is a commutation matrix if K

Usage

```
commutation_matrix(r, c)
```

Arguments

r	integer; row dimension.
c	integer; column dimension.

Examples

```
A <- randn(3, 4)
K <- commutation_matrix(3, 4)
compare <- function(x, y) {
    cbind(x, y, err = abs(x - y))
}
compare(K %*% vec(A), vec(t(A)))
```

commutation_matrix0 *Commutation matrix (memoised)*

Description

Commutation matrix (memoised)

Usage

```
commutation_matrix0(...)
```

Arguments

...	‘r‘ integer; row dimension. ‘c‘ integer; column dimension. Use empty argument to clear the cache.

cos,dual-method *Element-wise cosine of a dual object*

Description

Element-wise cosine of a dual object

Usage

```
## S4 method for signature 'dual'
cos(x)
```

Arguments

x	A "dual" object.

crossprod,dual,missing-method *Crossproduct of 'dual'-class objects*

Description

Crossproduct of 'dual'-class objects

Usage

```
## S4 method for signature 'dual,missing'
crossprod(x, y)

## S4 method for signature 'dual,ANY'
crossprod(x, y)

## S4 method for signature 'dual,dual'
crossprod(x, y)
```

Arguments

x	A numeric matrix, or the corresponding "dual" object.
y	A numeric matrix, or the corresponding "dual" object.

dchisq0 *The density of the chi-squared distribution*

Description

The density of the chi-squared distribution

Usage

```
dchisq0(x, df, log = FALSE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
df	degrees of freedom (> 0, maybe non-integer).
log	logical; if TRUE, returns the log value.

Examples

```
n <- 10
x <- rchisq0(n, df = 3)
dchisq0(x, df = 3)
```

det,dual-method *Determinant of a 'dual'-class object*

Description

Determinant of a 'dual'-class object

Usage

```
## S4 method for signature 'dual'
det(x, ...)
```

Arguments

x	A "dual" object.
...	Other parameters to be passed to 'base::det'.

det.dual *Determinant of a 'dual'-class object*

Description

Determinant of a 'dual'-class object

Usage

```
## S3 method for class 'dual'
det(x, ...)
```

Arguments

x	A "dual" object.
...	Other parameters to be passed to 'base::det'.

dexp0 *The density of the exponential distribution*

Description

The density of the exponential distribution

Usage

```
dexp0(x, rate = 1, log = FALSE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
rate	positive number; the rate parameter.
log	logical; if TRUE, returns the log value.

Examples

```
n <- 10
x <- rexp0(n, rate = 3)
dexp0(x, rate = 3)
```

dgamma0 *The density of the gamma distribution*

Description

The density of the gamma distribution

Usage

```
dgamma0(x, shape, rate, scale = 1/rate, log = FALSE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
shape	positive number; the shape parameter.
rate	positive number; the rate parameter.
scale	positive number; the scale parameter.
log	logical; if TRUE, returns the log value.

Examples

```
n <- 10
x <- rgamma0(n, shape = 1, scale = 5)
dgamma0(x, shape = 1, scale = 5)
```

diag.dual *Diagonal matrix*

Description

Diagonal matrix

Usage

```
diag.dual(x)

## S4 method for signature 'dual'
diag(x)
```

Arguments

x	A "dual" object.

Note

A single column matrix is treated as a vector. If one wants to call `diag` on a single column matrix `x`, one can call `x[1,1]` instead.

`Diagonal` *Diagonal matrix*

Description

Diagonal matrix

Usage

```
Diagonal(n, x = NULL)
```

Arguments

`n`	integer; the dimension of the square matrix.
`x`	A scalar or a vector to be placed on the diagonal of the matrix.

Examples

```
Diagonal(3)
Diagonal(3, 999)
Diagonal(3, c(11, 22, 33))
```

`Diagonal0` *Diagonal matrix (memoised)*

Description

Diagonal matrix (memoised)

Usage

```
Diagonal0(...)
```

Arguments

`...`	‘n‘ integer; the dimension of the square matrix. ‘x‘ A scalar or a vector to be placed on the diagonal of the matrix. Use empty argument to clear the cache.

dim,dual-method *Dimension of an Object*

Description

Dimension of an Object

Usage

```
## S4 method for signature 'dual'
dim(x)
```

Arguments

x	A "dual" object.

dinvWishart0 *The density of the inverse Wishart distribution*

Description

The density of the inverse Wishart distribution

Usage

```
dinvWishart0(X, v, M, log = FALSE)
```

Arguments

X	numeric matrix.
v	degrees of freedom (> 0, maybe non-integer).
M	numeric matrix; the scale matrix.
log	logical; if TRUE, returns the log value.

Examples

```
d <- 3
mat0 <- crossprod(randn(d, d))
X <- solve(rWishart0(v = 10, M = solve(mat0)))
dinvWishart0(X, v = 10, M = mat0)
```

dmvnorm0 *The density of the multivariate normal distribution*

Description

The density of the multivariate normal distribution

Usage

```
dmvnorm0(x, mean, sigma, log = FALSE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
mean	numeric vector; the mean vector.
sigma	numeric matrix; the covariance matrix.
log	logical; if TRUE, returns the log value.

Examples

```
n <- 10
d <- 2
sigma <- crossprod(randn(d, d))
x <- rmvnorm0(n, mean = numeric(d), sigma = sigma)
dmvnorm0(x, mean = numeric(d), sigma = sigma)
```

dmvt0 *The density of the multivariate t distribution*

Description

The density of the multivariate t distribution

Usage

```
dmvt0(x, delta, sigma, df = 1, log = TRUE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
delta	the vector of noncentrality parameters.
sigma	numeric matrix; scale matrix.
df	degress of freedom.
log	logical; whether to return log density value.

Examples

```
n <- 10
d <- 3
sigma <- crossprod(randn(d, d))
x <- rmvt0(n, sigma = sigma, df = 2)
dmvt0(x, delta = numeric(d), sigma = sigma, df = 2)
```

---

dnorm0 *The density of the normal distribution*

Description

The density of the normal distribution

Usage

```
dnorm0(x, mean, sd, log = FALSE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
mean	numeric vector; the mean vector.
sd	numeric vector; the sd vector.
log	logical; if TRUE, returns the log value.

Examples

```
n <- 10
x <- rnorm0(n, mean = 0, sd = 1)
dnorm0(x, mean = 0, sd = 1)
```

---

dt0 *The density of the student-t distribution*

Description

The density of the student-t distribution

Usage

```
dt0(x, df, log = FALSE)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
df	degrees of freedom (> 0, maybe non-integer).
log	logical; if TRUE, returns the log value.

Examples

```
n <- 10
x <- rt0(n, df = 3)
dt0(x, df = 3)
```

---

dual *Dual number constructor*

Description

Dual number constructor

Usage

```
dual(x, param_dim, ind)
```

Arguments

x	The object to be converted to a dual number.
param_dim	A named list. The dimension of the dual component to be attached.
ind	Integer; the index in ‘param_dim‘ corresponding to ‘x‘. Use ‘-1‘ if it is not applicable.

Value

A dual number with components "x" and "dx". The first gives the value of ‘f‘, and the second gives the derivative of ‘f‘.

Examples

```
# Suppose X is a 2 x 2 matrix, Y is a 3 x 1 vector, Z is a 2 x 3 matrix, and
# we wish to attach dual components {dX, dY, dZ} to X.
dual(randn(2, 2), list(X = 4, Y = 3, Z = 6), ind = 1)
```

dual-class *S4 class "dual"*

Description

This class attaches a dual component to a number / matrix.

Slots

`x` scalar, vector or matrix; also accepts any matrix classes from the "Matrix" package.

`dx` matrix; also accepts any matrix classes from the "Matrix" package.

`param` a named list, containing the column indices each variable occupies.

Note

Users should not construct the object directly, instead, use the constructor helper dual provided.

duals *Converting a list of parameters into a list of dual numbers*

Description

Converting a list of parameters into a list of dual numbers

Usage

```
duals(vary)
```

Arguments

`vary`	A named list of parameters

Value

A named list of dual numbers.

Examples

```
X <- randn(2, 2)
y <- rnorm(2)
duals(list(X = X, y = y))
```

dWishart0 *The density of the Wishart distribution*

Description

The density of the Wishart distribution

Usage

```
dWishart0(X, v, M, log = FALSE)
```

Arguments

X	numeric matrix.
v	degrees of freedom (> 0, maybe non-integer).
M	numeric matrix; the scale matrix.
log	logical; if TRUE, returns the log value.

Examples

```
d <- 3
mat0 <- crossprod(randn(d, d))
X <- rWishart0(v = 10, M = mat0)
dWishart0(X, v = 10, M = mat0)
```

elimination_matrix *Elimination matrix*

Description

Suppose A is a (n x n) symmetric matrix, then a (n(n-1)/2 x n^2) matrix E is an elimination matrix if E

Usage

```
elimination_matrix(n)
```

Arguments

n	integer, row or column dimension.

Examples

```
A <- crossprod(randn(3, 3))
E <- elimination_matrix(3)
compare <- function(x, y) {
    cbind(x, y, err = abs(x - y))
}
compare(E %*% vec(A), vech(A))
```

`elimination_matrix0` *Elimination matrix (memoised)*

Description

Elimination matrix (memoised)

Usage

```
elimination_matrix0(...)
```

Arguments

`...`	'n' integer, row or column dimension. Use empty argument to clear the cache.

`exp,dual-method` *Element-wise exponential of a dual object*

Description

Element-wise exponential of a dual object

Usage

```
## S4 method for signature 'dual'
exp(x)
```

Arguments

`x`	A "dual" object.

Extract.dual *Extract parts of an object*

Description

Extract parts of an object

Usage

```
Extract.dual(x, i, j, drop = FALSE)

## S4 method for signature 'dual,numeric,missing,ANY'
x[i, j, drop = FALSE]

## S4 method for signature 'dual,missing,numeric,ANY'
x[i, j, drop = FALSE]

## S4 method for signature 'dual,numeric,numeric,ANY'
x[i, j, drop = FALSE]
```

Arguments

x	A "dual" object.
i	integer; the row index.
j	integer; the column index.
drop	TRUE or FALSE; if TRUE, returns a vector when one dimension of the matrix is 1.

finite_diff *Finite difference method*

Description

Finite difference method

Usage

```
finite_diff(f, wrt = NULL, at, h = 1e-08, seed, method = "forward")
```

Arguments

f	A function of which the derivative is sought.
wrt	Character vector; the name of the variables to differentiate with respect to.
at	A named list of variables; the point at which the derivative is evaluated.
h	The finite differencing parameter; the size of perturbation.
seed	Seed; for pathwise derivative only.
method	"forward" for forward differencing or "central" for central differencing.

Value

A numeric matrix; the Jacobian matrix.

Examples

```
f <- function(y, X, beta) { y - X %*% beta }
finite_diff(
  f, wrt = "beta",
  at = list(X = matrix(1:4, 2, 2), y = c(2, 3), beta = c(5, 6))
)

g <- function(X, Y) { X %*% Y }
finite_diff(g, at = list(X = randn(2, 2), Y = randn(2, 2)))

h <- function(x) { exp(-x^2) }
finite_diff(h, at = list(x = 0.01), wrt = "x")
finite_diff(h, at = list(x = 0.01), wrt = "x", method = "central")
```

---

gamma,dual-method *Element-wise gamma of a dual object*

Description

Element-wise gamma of a dual object

Usage

```
## S4 method for signature 'dual'
gamma(x)
```

Arguments

x	A "dual" object.

head.dual *Return the First or Last Part of an Object*

Description

Return the First or Last Part of an Object

Usage

```
head.dual(x, n = 6)

## S4 method for signature 'dual'
head(x, n = 6)
```

Arguments

x	A "dual" object.
n	A single integer.

inverse_transform *Simulation by inverse transform*

Description

Simulation by inverse transform

Usage

```
inverse_transform(cdf)
```

Arguments

cdf	A function; the distribution function of a random variable.

IxCD *Compute (I %x% C) D*

Description

Compute (I %x% C) D

Usage

```
IxCD(C, D)
```

Arguments

C	numeric matrix.
D	numeric matrix.

kronecker,ANY,dual-method *Kronecker multiplication of 'dual'-class objects (ANY-dual)*

Description

Kronecker multiplication of 'dual'-class objects (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
kronecker(X, Y)
```

Arguments

X	"ANY" object.
Y	A "dual" object.

kronecker,dual,ANY-method

Kronecker multiplication of 'dual'-class objects (dual-ANY)

Description

Kronecker multiplication of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
kronecker(X, Y)
```

Arguments

X	A "dual" object.
Y	"ANY" object.

kronecker,dual,dual-method

Kronecker product of 'dual'-class objects (dual-dual)

Description

Kronecker product of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
kronecker(X, Y)
```

Arguments

X	A "dual" object.
Y	A "dual" object.

length,dual-method *Length of an Object*

Description

Length of an Object

Usage

```
## S4 method for signature 'dual'
length(x)
```

Arguments

x	A "dual" object.

log,dual-method *Element-wise logarithm of a dual object*

Description

Element-wise logarithm of a dual object

Usage

```
## S4 method for signature 'dual'
log(x)
```

Arguments

x	A "dual" object.

`lower_tri_matrix` *Construct a lower triangular matrix from a vector*

Description

Construct a lower triangular matrix from a vector

Usage

```
lower_tri_matrix(data, nrow = 1, ncol = 1, diag = FALSE)

## S4 method for signature 'dual'
lower_tri_matrix(data, nrow = 1, ncol = 1, diag = FALSE)
```

Arguments

`data`	A numeric vector.
`nrow`	A positive integer; the desired number of rows.
`ncol`	A positive integer; the desired number of rows.
`diag`	A logical; should the diagonal be included ?

Examples

```
lower_tri_matrix(1:3, 3, 3)
lower_tri_matrix(1:6, 3, 3, diag = TRUE)
```

`matrix` *Matrices*

Description

Matrices

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Arguments

data	A "dual" object.
nrow	a positive integer; the desired number of rows.
ncol	a positive integer; the desired number of rows.
byrow	TRUE or FALSE; whether to fill the matrix by rows.
dimnames	A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

---

matrix.dual *Coerce the first component of the dual object into a matrix.*

Description

Coerce the first component of the dual object into a matrix.

Usage

```
## S3 method for class 'dual'
matrix(data, nrow, ncol = 1, byrow = FALSE, dimnames = NULL)

## S4 method for signature 'dual'
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Arguments

data	A "dual" object.
nrow	a positive integer; the desired number of rows.
ncol	a positive integer; the desired number of rows.
byrow	TRUE or FALSE; whether to fill the matrix by rows.
dimnames	A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

matrix_determinant *Determinant of a matrix*

Description

Determinant of a matrix

Usage

```
det(x, ...)

det.default(x, ...)
```

Arguments

x	numeric matrix: logical matrices are coerced to numeric.
...	Optional arguments. At present none are used. Previous versions of det allowed an optional method argument. This argument will be ignored but will not produce an error.

matrix_prod *Interface for optimal matrix chain multiplication*

Description

Interface for optimal matrix chain multiplication

Usage

```
matrix_prod(..., method = "optimal")
```

Arguments

...	Numeric matrices.
method	"optimal" order or "natural" left-to-right order.

Examples

```
A <- randn(20, 5)
B <- randn(5, 40)
C <- randn(40, 2)
system.time({ matrix_prod(A, B, C, method = "optimal") })
system.time({ matrix_prod(A, B, C, method = "natural") })
```

`mcm_optimal_order` *Find the optimal order of multiplying a matrix chain*

Description

Find the optimal order of multiplying a matrix chain

Usage

```
mcm_optimal_order(x)
```

Arguments

`x`	A numeric vector of matrix dimensions

`mean,dual-method` *Mean of vector or matrix elements*

Description

Mean of vector or matrix elements

Usage

```
## S4 method for signature 'dual'
mean(x)
```

Arguments

`x`	A "dual" object.

`ncol,dual-method` *Number of columns*

Description

Number of columns

Usage

```
## S4 method for signature 'dual'
ncol(x)
```

Arguments

`x`	A "dual" object.

nrow,dual-method *Number of rows*

Description

Number of rows

Usage

```
## S4 method for signature 'dual'
nrow(x)
```

Arguments

x	A "dual" object.

one_matrix *Matrix of ones*

Description

Matrix of ones

Usage

```
one_matrix(nr, nc)
```

Arguments

nr	integer; row dimension.
nc	integer; column dimension.

Examples

```
one_matrix(1, 3)
one_matrix(5, 1)
one_matrix(2, 2)
```

one_matrix0 *Matrix of ones (memoised)*

Description

Matrix of ones (memoised)

Usage

```
one_matrix0(...)
```

Arguments

...	'nr' integer; number of rows. 'nc' integer; number of columns. Use empty argument to clear the cache.

optim_prod *Executing the matrix multiplication given the optimal order*

Description

Executing the matrix multiplication given the optimal order

Usage

```
optim_prod(matrix_ls)
```

Arguments

matrix_ls	A list of matrices to be multiplied

randn *Generate a matrix randomly from a normal distribution*

Description

Generate a matrix randomly from a normal distribution

Usage

```
randn(..., mean = 0, sd = 1)
```

Arguments

...	Any number of dimensions.
mean	vector of means.
sd	vector of standard deviations.

randu *Generate a matrix randomly from a uniform distribution*

Description

Generate a matrix randomly from a uniform distribution

Usage

```
randu(..., min = 0, max = 1)
```

Arguments

...	Any number of dimensions.
min	lower limit of the distribution. Must be finite.
max	upper limit of the distribution. Must be finite.

rbind2,ANY,dual-method

Combine 'dual'-class objects by Rows (ANY-dual)

Description

Combine 'dual'-class objects by Rows (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
rbind2(x, y)
```

Arguments

x	ANY object.
y	A "dual" object.

rbind2,dual,ANY-method

Combine 'dual'-class objects by Rows (dual-ANY)

Description

Combine 'dual'-class objects by Rows (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
rbind2(x, y)
```

Arguments

x	A "dual" object.
y	ANY object.

rbind2,dual,dual-method *Combine 'dual'-class objects by Rows (dual-dual)*

Description

Combine 'dual'-class objects by Rows (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
rbind2(x, y)
```

Arguments

x	A "dual" object.
y	A "dual" object.

rchisq0 *Simulate Chi-square random variates*

Description

Simulate Chi-square random variates

Usage

```
rchisq0(n, df, method = "inv_tf")
```

Arguments

n	Positive integer; number of observations.
df	Degrees of freedom (can be non-integer).
method	base or inv_tf; base refers to 'stats::rchisq' while inv_tf refers to inverse transform.

Examples

```
n <- 10
rchisq0(n, df = 3)
```

`rchisq0,numeric,dual-method` *Simulate Chi-square random variates*

Description

Simulate Chi-square random variates

Usage

```
## S4 method for signature 'numeric,dual'
rchisq0(n, df, method = "inv_tf")
```

Arguments

`n`	Positive integer; number of observations.
`df`	Degrees of freedom (can be non-integer).
`method`	base or inv_tf; base refers to the function in the 'stats' package while inv_tf refers to inverse transform.

`rexp0` *Simulate exponential random variates*

Description

Density, distribution function, quantile function and random generation for the exponential distribution with rate `rate` (i.e., mean `1/rate`).

Usage

```
rexp0(n, rate = 1)
```

Arguments

`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`rate`	vector of rates.

Details

If `rate` is not specified, it assumes the default value of `1`.

The exponential distribution with rate λ has density

$$f(x) = \lambda e^{-\lambda x}$$

for $x \geq 0$.

Value

dexp gives the density, pexp gives the distribution function, qexp gives the quantile function, and rexp generates random deviates.

The length of the result is determined by n for rexp, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Source

dexp, pexp and qexp are all calculated from numerically stable versions of the definitions.

rexp uses

Ahrens, J. H. and Dieter, U. (1972). Computer methods for sampling from the exponential and normal distributions. *Communications of the ACM*, **15**, 873–882.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 19. Wiley, New York.

See Also

exp for the exponential function.

Distributions for other standard distributions, including dgamma for the gamma distribution and dweibull for the Weibull distribution, both of which generalize the exponential.

Examples

```
dexp(1) - exp(-1) #-> 0

## a fast way to generate *sorted*  U[0,1]  random numbers:
rsunif <- function(n) { n1 <- n+1
   cE <- cumsum(rexp(n1)); cE[seq_len(n)]/cE[n1] }
plot(rsunif(1000), ylim=0:1, pch=".")
abline(0,1/(1000+1), col=adjustcolor(1, 0.5))
```

---

rexp0,numeric,dual-method

Simulate exponential random variates

Description

Simulate exponential random variates

Usage

```
## S4 method for signature 'numeric,dual'
rexp0(n, rate = 1)
```

Arguments

n	Positive integer; the number of samples.
rate	A dual number; the rate of the exponential distribution.

`rgamma0` *Simulate gamma random variates*

Description

Simulate gamma random variates

Usage

```
rgamma0(n, shape, scale, method = "inv_tf")
```

Arguments

n	Positive integer; the number of samples.
shape	Positive number; the shape of the gamma distribution.
scale	Positive number; the scale of the gamma distribution.
method	'base' or 'inv_tf'; 'base' refers to 'stats::rgamma' while 'inv_tf' refers to inverse transform.

Note

Inverse transform is slower, but it is provided to remedy that base R uses two algorithms to simulate gamma random variables in different parameter regions, which creates a discontinuity in the pathwise derivative.

Examples

```
n <- 10
rgamma0(n, shape = 1, scale = 1)
```

rgamma0,numeric,dual,dual-method *Simulate gamma random variates*

Description

Simulate gamma random variates

Usage

```
## S4 method for signature 'numeric,dual,dual'
rgamma0(n, shape, scale, method = "inv_tf")

## S4 method for signature 'numeric,dual,numeric'
rgamma0(n, shape, scale, method = "inv_tf")

## S4 method for signature 'numeric,numeric,dual'
rgamma0(n, shape, scale, method = "inv_tf")
```

Arguments

n	Positive integer; the number of samples.
shape	A dual number or a scalar; the shape of the gamma distribution.
scale	A dual number or a scalar; the scale of the gamma distribution.
method	'base' or 'inv_tf'; 'base' refers to 'stats::rgamma' while 'inv_tf' refers to inverse transform.

Note

At least one of 'shape' and 'scale' should be a dual number.

rmvnorm0 *Simulate multivariate normal random variates*

Description

Simulate multivariate normal random variates

Usage

```
rmvnorm0(n, mean, sigma)
```

Arguments

n	Positive integer; the number of samples.
mean	mean vector of the normal distribution.
sigma	covariance matrix of the normal distribution.

Value

A numeric matrix, where every column is a sample.

`rmvnorm0_dual` *Simulate multivariate normal random variates*

Description

Simulate multivariate normal random variates

Usage

```
rmvnorm0_dual(n, mean, sigma)

## S4 method for signature 'numeric,dual,dual'
rmvnorm0(n, mean, sigma)

## S4 method for signature 'numeric,dual,ANY'
rmvnorm0(n, mean, sigma)

## S4 method for signature 'numeric,ANY,dual'
rmvnorm0(n, mean, sigma)
```

Arguments

n	Positive integer; the number of samples.
mean	A numeric vector or a dual number; the mean of the normal distribution.
sigma	A numeric matrix or a dual number; the standard deviation of the normal distribution.

rmvt0 *Simulate random variates from the multivariate t distribution*

Description

Simulate random variates from the multivariate t distribution

Usage

```
rmvt0(n, sigma, df, delta = 0)
```

Arguments

n	positive integer; number of observations.
sigma	scale matrix.
df	degree of freedom.
delta	non-centrality parameters.

Examples

```
n <- 10
d <- 3
rmvt0(n, sigma = crossprod(randn(d, d)), df = 2)
```

rnorm0 *Simulate univariate normal random variates*

Description

Simulate univariate normal random variates

Usage

```
rnorm0(n, mean = 0, sd = 1)

## S4 method for signature 'numeric,dual,dual'
rnorm0(n, mean = 0, sd = 1)

## S4 method for signature 'numeric,ANY,dual'
rnorm0(n, mean = 0, sd = 1)

## S4 method for signature 'numeric,dual,ANY'
rnorm0(n, mean = 0, sd = 1)
```

Arguments

n	Positive integer; the number of samples.
mean	A dual number or a numeric vector; the mean of the normal distribution.
sd	A dual number or a numeric vector; the standard deviation of the normal distribution.

---

round,dual,integer-method *Rounding of Numbers*

Description

Rounding of Numbers

Usage

```
## S4 method for signature 'dual,integer'
round(x, digits = 0)
```

Arguments

x	A "dual" object.
digits	integer indicating the number of decimal places.

---

round.dual *Rounding of Numbers*

Description

Rounding of Numbers

Usage

```
## S3 method for class 'dual'
round(x, digits = 0)
```

Arguments

x	A "dual" object.
digits	An integer indicating the number of decimal places.

Note

The function 'round' does not have a derivative over the real line. The derivative will be kept unchanged. The reason of not dropping it is that sometimes one may need to round a matrix to correct floating-point errors. This is often used when an apparent symmetric matrix does not pass the symmetry test.

rowMeans,dual-method *Row mean of a matrix.*

Description

Row mean of a matrix.

Usage

```
## S4 method for signature 'dual'
rowMeans(x)
```

Arguments

x	A "dual" object.

rowSums,dual-method *Row sum of a matrix.*

Description

Row sum of a matrix.

Usage

```
## S4 method for signature 'dual'
rowSums(x)
```

Arguments

x	A "dual" object.

`rt0` *Simulate random variates from the student-t distribution*

Description

Simulate random variates from the student-t distribution

Usage

```
rt0(n, df)
```

Arguments

`n`	positive integer; the number of samples.
`df`	degrees of freedom (> 0, maybe non-integer). df = Inf is allowed.

Examples

```
n <- 10
rt0(10, df = 3)
```

`rt0,numeric,dual-method` *Simulate random variates from the student-t distribution*

Description

Simulate random variates from the student-t distribution

Usage

```
## S4 method for signature 'numeric,dual'
rt0(n, df)
```

Arguments

`n`	positive integer; the number of samples.
`df`	A dual number; the degree of freedom.

rWishart0 *Simulate Wishart random variates*

Description

Simulate Wishart random variates

Usage

```
rWishart0(v, M, method = "inv_tf")
```

Arguments

v	A scalar; degrees of freedom.
M	A matrix; the matrix parameter of the Wishart distribution.
method	base or inv_tf; base refers to the function in the 'stats' package while inv_tf refers to inverse transform.

Examples

```
d <- 4
rWishart0(3, crossprod(randn(d, d)))
```

rWishart0,dual,dual-method
Simulating from Wishart distribution using Bartlett decomposition

Description

Simulating from Wishart distribution using Bartlett decomposition

Usage

```
## S4 method for signature 'dual,dual'
rWishart0(v, M, method = "inv_tf")

## S4 method for signature 'ANY,dual'
rWishart0(v, M, method = "inv_tf")

## S4 method for signature 'dual,ANY'
rWishart0(v, M, method = "inv_tf")
```

Arguments

v	A dual number or a scalar; degrees of freedom.
M	A dual number or a matrix; the matrix parameter of the Wishart distribution.
method	base or inv_tf; base refers to the function in the ‘stats‘ package while inv_tf refers to inverse transform.

---

sin,dual-method *Element-wise sine of a dual object*

Description

Element-wise sine of a dual object

Usage

```
## S4 method for signature 'dual'
sin(x)
```

Arguments

x	A "dual" object.

---

solve,dual,missing-method *Inverse of 'dual'-class objects*

Description

Inverse of 'dual'-class objects

Usage

```
## S4 method for signature 'dual,missing'
solve(a, b, ...)

## S4 method for signature 'dual,dual'
solve(a, b, ...)

## S4 method for signature 'ANY,dual'
solve(a, b, ...)

## S4 method for signature 'dual,ANY'
solve(a, b, ...)
```

Arguments

a	A numeric square matrix or a corresponding "dual" object.
b	A numeric vector or matrix or a corresponding "dual" object.
...	Other arguments passed to 'base::solve'. See '?solve' for detail.

Note

At least one of 'a' and 'b' must be a dual object, and the other one may be any appropriate numeric matrix.

---

sqrt,dual-method *Element-wise square-root of a dual object*

Description

Element-wise square-root of a dual object

Usage

```
## S4 method for signature 'dual'
sqrt(x)
```

Arguments

x	A "dual" object.

---

sum,dual-method *Sum of matrix elements*

Description

Sum of matrix elements

Usage

```
## S4 method for signature 'dual'
sum(x)
```

Arguments

x	A "dual" object.

t.dual *Transpose of 'dual'-class objects*

Description

Transpose of 'dual'-class objects

Usage

```
## S3 method for class 'dual'
t(x)

## S4 method for signature 'dual'
t(x)
```

Arguments

x	A "dual" object.

tail.dual *Return the First or Last Part of an Object*

Description

Return the First or Last Part of an Object

Usage

```
tail.dual(x, n = 6)

## S4 method for signature 'dual'
tail(x, n = 6)
```

Arguments

x	A "dual" object.
n	A single integer.

tan,dual-method *Element-wise tangent of a dual object*

Description

Element-wise tangent of a dual object

Usage

```
## S4 method for signature 'dual'
tan(x)
```

Arguments

x	A "dual" object.

tcrossprod,dual,ANY-method *Crossproduct of 'dual'-class objects*

Description

Crossproduct of 'dual'-class objects

Usage

```
## S4 method for signature 'dual,ANY'
tcrossprod(x, y)

## S4 method for signature 'dual,dual'
tcrossprod(x, y)

## S4 method for signature 'dual,missing'
tcrossprod(x, y)
```

Arguments

x	A numeric matrix, or the corresponding "dual" object.
y	A numeric matrix, or the corresponding "dual" object.

tidy_dx *Add rownames and colnames to the dual component*

Description

Add rownames and colnames to the dual component

Usage

```
tidy_dx(x_dual, vary)
```

Arguments

x_dual	A dual number
vary	A named list of parameters

tr *Trace of a matrix*

Description

Trace of a matrix

Usage

```
tr(x)
```

Arguments

x	A square numeric matrix

Examples

```
A <- randn(2, 2)
tr(A)

B <- randn(3, 3)
tr(B)
```

tr,dual-method *Trace of a matrix*

Description

Trace of a matrix

Usage

```
## S4 method for signature 'dual'
tr(x)
```

Arguments

x	A "dual" object.

upper_tri_matrix *Construct an upper triangular matrix from a vector*

Description

Construct an upper triangular matrix from a vector

Usage

```
upper_tri_matrix(data, nrow = 1, ncol = 1, diag = FALSE)

## S4 method for signature 'dual'
upper_tri_matrix(data, nrow = 1, ncol = 1, diag = FALSE)
```

Arguments

data	A numeric vector.
nrow	A positive integer; the desired number of rows.
ncol	A positive integer; the desired number of rows.
diag	A logical; should the diagonal be included ?

Examples

```
upper_tri_matrix(1:3, 3, 3)
upper_tri_matrix(1:6, 3, 3, diag = TRUE)
```

vec *Vectorisation*

Description

Vectorisation

Usage

```
vec(x)
```

Arguments

x	A matrix.

Examples

```
A <- randn(3, 3)
vec(A)
```

vec,dual-method *Vectorisation*

Description

Vectorisation

Usage

```
## S4 method for signature 'dual'
vec(x)
```

Arguments

x	A "dual" object.

vech *Half-vectorisation*

Description

Half-vectorisation

Usage

```
vech(x)
```

Arguments

x	A matrix.

Examples

```
A <- randn(3, 3)
vech(A)
```

vech,dual-method *Half-vectorisation*

Description

Half-vectorisation

Usage

```
## S4 method for signature 'dual'
vech(x)
```

Arguments

x	A "dual" object.

XBxA *Pre-multiplying a kronecker product*

Description

Compute X (B %x% A)

Usage

```
XBxA(X, B, A)
```

Arguments

X	numeric matrix.
B	numeric matrix.
A	numeric matrix.

zero_matrix *Matrix of zeroes*

Description

Matrix of zeroes

Usage

```
zero_matrix(nr, nc)
```

Arguments

nr	integer; row dimension.
nc	integer; column dimension.

Examples

```
zero_matrix(1, 3)
zero_matrix(5, 1)
zero_matrix(2, 2)
```

zero_matrix0 *Matrix of zeroes (memoised)*

Description

Matrix of zeroes (memoised)

Usage

```
zero_matrix0(...)
```

Arguments

...	'nr' integer; number of rows. 'nc' integer; number of columns. Use empty argument to clear the cache.

%*%,ANY,dual-method *Matrix multiplication of 'dual'-class objects (ANY-dual)*

Description

Matrix multiplication of 'dual'-class objects (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
x %*% y
```

Arguments

x	"ANY" object.
y	A "dual" object.

%*%,dual,ANY-method *Matrix multiplication of 'dual'-class objects (dual-ANY)*

Description

Matrix multiplication of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
x %*% y
```

Arguments

x	A "dual" object.
y	"ANY" object.

%*%,dual,dual-method *Matrix multiplication of 'dual'-class objects (dual-dual)*

Description

Matrix multiplication of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
x %*% y
```

Arguments

x	A "dual" object.
y	A "dual" object.

^,ANY,dual-method *Powers of 'dual'-class objects (ANY-dual)*

Description

Powers of 'dual'-class objects (ANY-dual)

Usage

```
## S4 method for signature 'ANY,dual'
e1 ^ e2
```

Arguments

e1	"ANY" object.
e2	A "dual" object.

^,dual,ANY-method *Powers of 'dual'-class objects (dual-ANY)*

Description

Powers of 'dual'-class objects (dual-ANY)

Usage

```
## S4 method for signature 'dual,ANY'
e1 ^ e2
```

Arguments

e1	A "dual" object.
e2	"ANY" object.

`^,dual,dual-method` *Powers of 'dual'-class objects (dual-dual)*

Description

Powers of 'dual'-class objects (dual-dual)

Usage

```
## S4 method for signature 'dual,dual'
e1 ^ e2
```

Arguments

`e1`	A "dual" object.
`e2`	A "dual" object.

Index