Package: nabla (via r-universe)

October 28, 2026

Title Exact Derivatives via Automatic Differentiation

Version 0.7.2

URL https://github.com/queelius/nabla, https://metafunctor.com

BugReports https://github.com/queelius/nabla/issues

Description Automatic differentiation for R functions written with supported scalar arithmetic and mathematical operations. Provides a composable derivative operator D that computes gradients, Hessians, Jacobians, and arbitrary-order derivative tensors at machine precision. D(D(f)) gives Hessians, D(D(D(f))) gives third-order tensors for skewness of maximum likelihood estimators, and so on to higher orders. Supports loops and executed branches composed of supported operations. Uses nested dual numbers without finite difference truncation error; floating-point rounding and conditioning still affect results.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0), methods

Imports stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Collate 'dual-class.R' 'dual-arithmetic.R' 'dual-math.R' 'dual-special.R' 'dual-higher.R' 'derivatives.R' 'nabla-package.R'

NeedsCompilation no

Author Alexander Towell [aut, cre] (ORCID: <https://orcid.org/0000-0001-6443-9897>)

Maintainer Alexander Towell <queelius@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-28 11:20:08 UTC

RemoteUrl https://github.com/cran/nabla

RemoteRef HEAD

RemoteSha 2c225a419a9b9a1adc2f9d65d413f7b5b1e0742b

Contents

nabla-package . . . 3
beta . . . 4
D . . . 5
deriv . . . 6
deriv_n . . . 6
differentiate_n . . . 7
dual . . . 8
dual-arithmetic . . . 8
dual-atan2 . . . 10
dual-coerce . . . 11
dual-combine . . . 12
dual-is-numeric . . . 12
dual-log . . . 13
dual-math . . . 14
dual-math2 . . . 15
dual-maxmin . . . 15
dual-show . . . 16
dual-summary . . . 17
dual_constant . . . 17
dual_constant_n . . . 18
dual_variable . . . 19
dual_variable_n . . . 19
dual_vector . . . 20
dual_vector-access . . . 20
dual_vector-class . . . 21
dualr-class . . . 21
erf . . . 22
erfc . . . 22
gradient . . . 23
hessian . . . 24
is_dual . . . 24
jacobian . . . 25
lbeta . . . 26
psigamma . . . 27
value . . . 27

Index **29**

nabla-package *nabla: Exact Derivatives via Automatic Differentiation*

Description

Implements forward-mode automatic differentiation using dual numbers with S4 classes. Supports exact arbitrary-order derivatives through recursive nesting of duals, with high-level functions for computing gradients, Hessian matrices, and Jacobians of functions composed of supported scalar operations. Loops and executed branches are supported; arbitrary compiled functions and matrix operations are not.

Core Types

`dual` Constructor for dual numbers.

`dual_variable` Shorthand for `dual(x, 1)`.

`dual_constant` Shorthand for `dual(x, 0)`.

`dual_vector` Container for indexable dual vectors.

Accessors

`value` Extract the primal value.

`deriv` Extract the derivative component.

Higher-Order Derivatives

`dual_variable_n` Create a dual seeded for n-th order differentiation.

`deriv_n` Extract the k-th derivative from a nested dual result.

`differentiate_n` Compute f(x) and all derivatives up to order n.

Multi-Parameter Derivatives

`D` Composable total derivative operator. `D(f)` returns the derivative function; apply k times for k-th order tensors.

`gradient` Gradient of a scalar-valued function.

`hessian` Hessian matrix of a scalar-valued function.

`jacobian` Jacobian matrix of a vector-valued function.

Author(s)

Maintainer: Alexander Towell <queelius@gmail.com> (ORCID)

References

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153), 1–43.

See Also

Related CRAN packages: **dual**, **numDeriv**, **madness**

`beta` *Beta function for dual numbers*

Description

Computed as `exp(lbeta(a, b))`.

Usage

```
beta(a, b)

## S4 method for signature 'numeric,numeric'
beta(a, b)

## S4 method for signature 'ANY,ANY'
beta(a, b)
```

Arguments

`a`	A numeric or dual value.
`b`	A numeric or dual value.

Value

`beta(a, b)` with derivative.

Examples

```
beta(2, 3)
a <- dual_variable(2)
value(beta(a, 3))
```

D *Composable total derivative operator*

Description

`D(f)` returns the derivative of `f` as a new function. `D(f, x)` evaluates the derivative at `x`. `D(D(f))` composes for second-order derivatives, and so on.

Usage

```
D(f, x = NULL, order = 1L)
```

Arguments

`f`	A function taking a parameter vector (via `x[i]` indexing) and returning a scalar, list, or `dual_vector`.
`x`	Optional numeric vector. If provided, evaluates `D(f)(x)`.
`order`	Derivative order (default 1). `order = k` applies `D` k times.

Details

Each application of `D` appends one `n`-dimension to the output shape, where `n = length(x)`:

- For `f: R^n -> R`: D gives `(n)` gradient, D^2 gives `(n,n)` Hessian, D^3 gives `(n,n,n)`, etc.
- For `f: R^n -> R^m`: D gives `(m,n)` Jacobian, D^2 gives `(m,n,n)`, etc.

The composability works because the S4 dispatch for `dualr` arithmetic handles nested duals recursively. When `D(f)` is called with dual inputs (from an outer `D`), derivative propagation is automatic.

`gradient()`, `hessian()`, and `jacobian()` are convenience wrappers: `gradient(f, x)` is `D(f, x)`, `hessian(f, x)` is `D(f, x, order = 2)`, and `jacobian(f, x)` is `D(f, x)`.

Value

If `x` is `NULL`, a function. Otherwise, a numeric vector, matrix, or array of the appropriate tensor shape.

Examples

```
f <- function(x) x[1]^2 * x[2]
D(f, c(3, 4))
D(f, c(3, 4), order = 2)

g <- function(x) list(x[1] * x[2], x[1]^2)
D(g, c(2, 3))

Df <- D(f)
DDf <- D(Df)
DDf(c(3, 4))
```

deriv *Extract the derivative (tangent) part of a dual number*

Description

Extract the derivative (tangent) part of a dual number

Usage

```
deriv(d)

## S4 method for signature 'dualr'
deriv(d)

## S4 method for signature 'numeric'
deriv(d)
```

Arguments

d	A dual object.

Value

The deriv slot.

Examples

```
deriv(dual(3, 1))
```

deriv_n *Extract the k-th derivative from a nested dual result*

Description

After evaluating a function on a dual created by dual_variable_n, use deriv_n to extract any derivative from 0 (the function value) up to the seeded order.

Usage

```
deriv_n(d, k)
```

Arguments

d	A (possibly nested) dual number, or a numeric.
k	A non-negative integer: 0 for the function value, 1 for the first derivative, etc.

Value

A numeric value.

Examples

```
x <- dual_variable_n(1, order = 3)
r <- exp(x)
deriv_n(r, 0)
deriv_n(r, 1)
deriv_n(r, 2)
deriv_n(r, 3)
```

`differentiate_n` *Compute a function value and all derivatives up to order n*

Description

Evaluates `f` at a dual variable seeded for order `n`, returning the function value and all derivatives from 1 to `n`.

Usage

```
differentiate_n(f, x, order)
```

Arguments

`f`	A function of one numeric argument.
`x`	A numeric value at which to differentiate.
`order`	A positive integer: the maximum derivative order.

Value

A named list with components `value`, `d1`, `d2`, ..., `d<order>`.

Examples

```
differentiate_n(sin, pi/4, order = 4)
```

dual *Create a dual number*

Description

Create a dual number

Usage

```
dual(value, deriv = 0)
```

Arguments

value	The primal value (numeric or dual for nesting).
deriv	The derivative component (numeric or dual for nesting). Defaults to 0.

Value

A dual object.

Examples

```
x <- dual(3, 1)
value(x)
deriv(x)
```

dual-arithmetic *Arithmetic and comparison operators for dual numbers*

Description

Implements arithmetic (+, -, *, /, ^), comparison (==, !=, <, >, <=, >=), and logical (&, |) operators. Derivatives follow standard calculus rules (sum, product, quotient, power, chain).

Usage

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

## S4 method for signature 'dualr,numeric'
e1 + e2

## S4 method for signature 'numeric,dualr'
e1 + e2

## S4 method for signature 'dualr,dualr'
```

```
e1 - e2

## S4 method for signature 'dualr,numeric'
e1 - e2

## S4 method for signature 'numeric,dualr'
e1 - e2

## S4 method for signature 'dualr,dualr'
e1 * e2

## S4 method for signature 'dualr,numeric'
e1 * e2

## S4 method for signature 'numeric,dualr'
e1 * e2

## S4 method for signature 'dualr,dualr'
e1 / e2

## S4 method for signature 'dualr,numeric'
e1 / e2

## S4 method for signature 'numeric,dualr'
e1 / e2

## S4 method for signature 'dualr,dualr'
e1 ^ e2

## S4 method for signature 'dualr,numeric'
e1 ^ e2

## S4 method for signature 'numeric,dualr'
e1 ^ e2

## S4 method for signature 'dualr,dualr'
Ops(e1, e2)

## S4 method for signature 'dualr,numeric'
Ops(e1, e2)

## S4 method for signature 'numeric,dualr'
Ops(e1, e2)

## S4 method for signature 'dualr,missing'
e1 + e2

## S4 method for signature 'dualr,missing'
```

```
e1 - e2

## S4 method for signature 'dualr'
!x
```

Arguments

e1, e2	Dual or numeric operands.
x	A dual number (for unary !).

Value

A dual for arithmetic ops; logical for comparisons.

Examples

```
x <- dual_variable(3)
y <- dual_variable(4)

value(x + y)
deriv(x * x)
value(x^2)
deriv(x^2)

x < y
x == y
```

---

dual-atan2 *Two-argument arctangent for dual numbers*

Description

Two-argument arctangent for dual numbers

Usage

```
## S4 method for signature 'dualr,dualr'
atan2(y, x)

## S4 method for signature 'dualr,numeric'
atan2(y, x)

## S4 method for signature 'numeric,dualr'
atan2(y, x)
```

Arguments

y	A dual or numeric.
x	A dual or numeric.

Value

A dual representing atan2(y, x) with correct derivative.

Examples

```
y <- dual_variable(1)
x <- dual_constant(1)
result <- atan2(y, x)
value(result)
```

---

dual-coerce *Coerce dual to numeric*

Description

Extracts the primal value, discarding the derivative.

Usage

```
## S4 method for signature 'dualr'
as.numeric(x, ...)
```

Arguments

x	A dual object.
...	Ignored.

Value

Numeric value.

Examples

```
x <- dual(3.14, 1)
as.numeric(x)
```

dual-combine *Combine dual numbers into a dual_vector*

Description

Combine dual numbers into a dual_vector

Usage

```
## S4 method for signature 'dualr'
c(x, ..., recursive = FALSE)
```

Arguments

x	A dual number.
...	Additional duals or numerics.
recursive	Ignored.

Value

A dual_vector.

Examples

```
x <- dual_variable(1)
y <- dual_variable(2)
dv <- c(x, y)
length(dv)
```

dual-is-numeric *Check if a dual number is numeric*

Description

Returns TRUE for dual numbers so that defensive type checks pass.

Usage

```
## S4 method for signature 'dualr'
is.numeric(x)
```

Arguments

x	A dual object.

Value

TRUE.

Examples

```
is.numeric(dual(1, 0))
```

---

dual-log *Logarithm with optional base for dual numbers*

Description

Logarithm with optional base for dual numbers

Usage

```
## S4 method for signature 'dualr'
log(x, base = exp(1))
```

Arguments

x	A dual number.
base	Numeric base (default: exp(1) for natural log).

Value

A dual representing log(x, base).

Examples

```
x <- dual_variable(8)
value(log(x, base = 2))
deriv(log(x, base = 2))
```

dual-math *Math group generic for dual numbers*

Description

Implements all standard mathematical functions for dual numbers via the chain rule: `f(dual(a, b)) = dual(f(a), df(a) * b)`.

Supported functions: `abs`, `sign`, `sqrt`, `floor`, `ceiling`, `trunc`, `round`, `exp`, `expm1`, `log`, `log2`, `log10`, `log1p`, `cos`, `sin`, `tan`, `cospi`, `sinpi`, `tanpi`, `acos`, `asin`, `atan`, `cosh`, `sinh`, `tanh`, `acosh`, `asinh`, `atanh`, `gamma`, `lgamma`, `digamma`, `trigamma`, `cumsum`, `factorial`, `lfactorial`.

Usage

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

## S4 method for signature 'dualr'
sqrt(x)

## S4 method for signature 'dualr'
Math(x)
```

Arguments

`x`	A `dual` number.

Value

A `dual` with the function applied to the value and the derivative propagated via the chain rule.

Examples

```
x <- dual_variable(pi / 4)
value(sin(x))
deriv(sin(x))

y <- dual_variable(2)
value(exp(y))
deriv(exp(y))
deriv(log(y))
```

dual-math2 *Math2 group generic for dual numbers*

Description

Implements `round` and `signif` for dual numbers. These are piecewise constant functions, so the derivative is zero almost everywhere.

Usage

```
## S4 method for signature 'dualr'
Math2(x, digits)
```

Arguments

`x`	A `dual` number.
`digits`	Integer; number of digits for rounding.

Value

A `dual` with the rounded value and zero derivative.

Examples

```
x <- dual_variable(3.14159)
value(round(x, 2))
deriv(round(x, 2))
```

dual-maxmin *Piecewise max and min for dual numbers*

Description

Compares on value and propagates the derivative of the selected branch.

Usage

```
## S4 method for signature 'dualr'
max(x, ..., na.rm = FALSE)

## S4 method for signature 'dualr'
min(x, ..., na.rm = FALSE)
```

Arguments

x	A dual number.
...	Additional dual or numeric values.
na.rm	Ignored.

Value

A dual representing the max or min.

Examples

```
x <- dual_variable(3)
y <- dual_variable(5)
value(max(x, y))
value(min(x, y))
```

---

dual-show *Display a dual number*

Description

Display a dual number

Usage

```
## S4 method for signature 'dualr'
show(object)

## S4 method for signature 'dual_vector'
show(object)
```

Arguments

object	A dual object.

Value

Invisible NULL; called for side effect of printing.

Examples

```
x <- dual(3, 1)
x

dv <- dual_vector(dual(1, 0), dual(2, 1))
dv
```

dual-summary *Summary group generic for dual numbers*

Description

Implements sum, prod, min, max, range, any, and all for dual numbers. Derivatives are propagated correctly through sum (additive) and prod (multiplicative).

Usage

```
## S4 method for signature 'dualr'
Summary(x, ..., na.rm = FALSE)
```

Arguments

x	A dual number.
...	Additional dual or numeric values.
na.rm	Logical; ignored (present for generic compatibility).

Value

A dual for sum/prod/min/max; a dual_vector for range; logical for any/all.

Examples

```
x <- dual_variable(2)
y <- dual_variable(5)
value(sum(x, y))
value(prod(x, y))
```

dual_constant *Create a dual constant (derivative seed = 0)*

Description

Wraps a numeric value as a dual with zero derivative, representing a constant with respect to the differentiation variable.

Usage

```
dual_constant(x)
```

Arguments

x	A numeric value.

Value

A `dual` with `value = x` and `deriv = 0`.

Examples

```
k <- dual_constant(5)
deriv(k)
```

`dual_constant_n` *Create a constant dual for n-th order differentiation*

Description

Wraps a numeric value as a nested dual with all derivative components zero, representing a constant with respect to the differentiation variable at nesting depth `n`.

Usage

```
dual_constant_n(x, order)
```

Arguments

`x`	A numeric value.
`order`	A non-negative integer specifying the nesting depth.

Value

A (possibly nested) dual number with zero derivatives.

Examples

```
k <- dual_constant_n(5, order = 3)
deriv_n(k, 1)
deriv_n(k, 2)
deriv_n(k, 3)
```

dual_variable *Create a dual variable (derivative seed = 1)*

Description

Convenience constructor for the independent variable when computing derivatives. Sets `deriv = 1` so that the output's derivative slot contains df/dx.

Usage

```
dual_variable(x)
```

Arguments

`x`	A numeric value.

Value

A `dual` with `value = x` and `deriv = 1`.

Examples

```
x <- dual_variable(2)
deriv(x^2)
```

dual_variable_n *Create a dual seeded for n-th order differentiation*

Description

Recursively nests dual numbers to enable exact computation of derivatives up to order `n`. The variable is seeded so that after evaluating a function `f`, the k-th derivative can be extracted with `deriv_n(result, k)`.

Usage

```
dual_variable_n(x, order)
```

Arguments

`x`	A numeric value at which to differentiate.
`order`	A positive integer specifying the derivative order.

Value

A (possibly nested) dual number.

Examples

```
x <- dual_variable_n(2, order = 3)
r <- x^4
deriv_n(r, 3)
```

`dual_vector` *Create a vector of dual numbers*

Description

Wraps a list of dual objects in a container that supports `[]` indexing and `length()`, so that user functions can use natural `theta[1]` notation.

Usage

```
dual_vector(...)
```

Arguments

`...`	Dual objects, or a single list of dual objects.

Value

A `dual_vector`.

Examples

```
dv <- dual_vector(dual(1, 0), dual(2, 1))
length(dv)
value(dv[1])
```

`dual_vector-access` *Indexing and length for dual_vector*

Description

Indexing and length for dual_vector

Usage

```
## S4 method for signature 'dual_vector,numeric'
x[i, j, ..., drop = TRUE]

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

Arguments

x	A dual_vector.
i	Numeric index.
j, drop, ...	Ignored (present for generic compatibility).

Value

A single dual for scalar index; a dual_vector for vector index; an integer for length.

Examples

```
dv <- dual_vector(dual(10, 1), dual(20, 0), dual(30, 0))
value(dv[1])
length(dv)
```

---

dual_vector-class *Dual Number Vector*

Description

A container for multiple dual numbers that supports indexing with [and [[, allowing log-likelihood functions to be written with theta[1], theta[2] notation.

Slots

.Data List of dual objects.

---

dualr-class *Dual Number Class for Automatic Differentiation*

Description

S4 class representing a dual number $a + b\varepsilon$ where $\varepsilon^2 = 0$. The value slot holds the primal value and the deriv slot holds the tangent (derivative) component. Both slots accept ANY type to support nested duals for higher-order derivatives.

Slots

value The primal (function) value. Numeric for first-order duals, or another dual for higher-order.

deriv The tangent (derivative) component. Numeric for first-order duals, or another dual for higher-order.

erf *Error function*

Description

Computes $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

Usage

```
erf(x)

## S4 method for signature 'numeric'
erf(x)

## S4 method for signature 'dualr'
erf(x)
```

Arguments

x	A numeric or dual value.

Value

The error function value. For dual input, returns a dual with derivative $(2/\sqrt{\pi})e^{-x^2}$.

Examples

```
erf(1)
x <- dual_variable(1)
value(erf(x))
```

erfc *Complementary error function*

Description

Computes $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$.

Usage

```
erfc(x)

## S4 method for signature 'numeric'
erfc(x)

## S4 method for signature 'dualr'
erfc(x)
```

Arguments

x	A numeric or dual value.

Value

The complementary error function value.

Examples

```
erfc(1)
x <- dual_variable(0)
value(erfc(x))
```

---

gradient *Compute the gradient of a scalar-valued function*

Description

Evaluates the gradient of f at x using forward-mode AD. Equivalent to D(f, x) for scalar-valued f.

Usage

```
gradient(f, x)
```

Arguments

f	A function taking a parameter vector (numeric or dual) and returning a scalar. Parameters are accessed via x[i].
x	A numeric vector of parameter values.

Details

The function should use x[1], x[2], etc. to access parameters. This is compatible with the standard R convention used by optim.

Value

A numeric vector of length p containing the gradient.

Examples

```
f <- function(x) -(x[1] - 3)^2 - (x[2] - 5)^2
gradient(f, c(1, 2))
```

hessian *Compute the Hessian of a scalar-valued function*

Description

Computes the matrix of second partial derivatives of `f` at `x` using forward-mode AD. Equivalent to `D(f, x, order = 2)`.

Usage

```
hessian(f, x)
```

Arguments

`f`	A function taking a parameter vector and returning a scalar.
`x`	A numeric vector of parameter values.

Value

A `p` x `p` numeric matrix (the Hessian).

Examples

```
f <- function(x) -(x[1] - 3)^2 - (x[2] - 5)^2
hessian(f, c(1, 2))
```

is_dual *Test whether an object is a dual number*

Description

Test whether an object is a dual number

Usage

```
is_dual(x)
```

Arguments

`x`	An object.

Value

Logical.

Examples

```
is_dual(dual(1, 0))
is_dual(42)
```

`jacobian` *Compute the Jacobian of a vector-valued function*

Description

Computes the first derivative of `f` at `x` using forward-mode AD. Equivalent to `D(f, x)`. For `f: R^p -> R^m`, returns an `m x p` matrix. For scalar-valued `f`, returns a length-p gradient vector.

Usage

```
jacobian(f, x)
```

Arguments

`f`	A function taking a parameter vector and returning a scalar, list of scalars, or a `dual_vector`.
`x`	A numeric vector of parameter values (length `p`).

Value

An `m x p` numeric matrix for vector-valued `f`, or a numeric vector of length `p` for scalar-valued `f`.

Examples

```
f <- function(x) {
  a <- x[1]; b <- x[2]
  list(a * b, a^2, sin(b))
}
jacobian(f, c(2, pi/4))

g <- function(x) x[1]^2 + x[2]^2
jacobian(g, c(3, 4))
```

lbeta *Log-beta function for dual numbers*

Description

Log-beta function for dual numbers

Usage

```
lbeta(a, b)

## S4 method for signature 'numeric,numeric'
lbeta(a, b)

## S4 method for signature 'dualr,dualr'
lbeta(a, b)

## S4 method for signature 'dualr,numeric'
lbeta(a, b)

## S4 method for signature 'numeric,dualr'
lbeta(a, b)
```

Arguments

a	A numeric or dual value.
b	A numeric or dual value.

Value

`lbeta(a, b)` with derivative via digamma.

Examples

```
lbeta(2, 3)
a <- dual_variable(2)
value(lbeta(a, 3))
```

psigamma *Polygamma function for dual numbers*

Description

Computes $\psi^{(n)}(x)$ where n is the derivative order. The derivative of $\psi^{(n)}(x)$ is $\psi^{(n+1)}(x)$.

Usage

```
psigamma(x, deriv = 0L)

## S4 method for signature 'numeric'
psigamma(x, deriv = 0L)

## S4 method for signature 'dualr'
psigamma(x, deriv = 0L)
```

Arguments

x	A numeric or dual value.
deriv	Integer derivative order (0 = digamma, 1 = trigamma, etc.).

Value

The polygamma function value.

Examples

```
psigamma(1, deriv = 0)
x <- dual_variable(2)
value(psigamma(x, deriv = 1))
```

value *Extract the value (primal) part of a dual number*

Description

Extract the value (primal) part of a dual number

Usage

```
value(d)

## S4 method for signature 'dualr'
value(d)

## S4 method for signature 'numeric'
value(d)
```

Arguments

d	A dual object.

Value

The value slot.

Examples

```
value(dual(3, 1))
```

Index

!,dualr-method *(dual-arithmetic)*, 8
*,dualr,dualr-method *(dual-arithmetic)*, 8
*,dualr,numeric-method *(dual-arithmetic)*, 8
*,numeric,dualr-method *(dual-arithmetic)*, 8
+,dualr,dualr-method *(dual-arithmetic)*, 8
+,dualr,missing-method *(dual-arithmetic)*, 8
+,dualr,numeric-method *(dual-arithmetic)*, 8
+,numeric,dualr-method *(dual-arithmetic)*, 8
-,dualr,dualr-method *(dual-arithmetic)*, 8
-,dualr,missing-method *(dual-arithmetic)*, 8
-,dualr,numeric-method *(dual-arithmetic)*, 8
-,numeric,dualr-method *(dual-arithmetic)*, 8
/,dualr,dualr-method *(dual-arithmetic)*, 8
/,dualr,numeric-method *(dual-arithmetic)*, 8
/,numeric,dualr-method *(dual-arithmetic)*, 8
[,dual_vector,numeric-method *(dual_vector-access)*, 20
^,dualr,dualr-method *(dual-arithmetic)*, 8
^,dualr,numeric-method *(dual-arithmetic)*, 8
^,numeric,dualr-method *(dual-arithmetic)*, 8

as.numeric,dualr-method *(dual-coerce)*, 11
atan2,dualr,dualr-method *(dual-atan2)*, 10
atan2,dualr,numeric-method *(dual-atan2)*, 10
atan2,numeric,dualr-method *(dual-atan2)*, 10

beta, 4
beta,ANY,ANY-method *(beta)*, 4
beta,numeric,numeric-method *(beta)*, 4

c,dualr-method *(dual-combine)*, 12

D, *3*, 5
deriv, *3*, 6
deriv,dualr-method *(deriv)*, 6
deriv,numeric-method *(deriv)*, 6
deriv_n, *3*, 6
differentiate_n, *3*, 7
dual, *3*, 8
dual-arithmetic, 8
dual-atan2, 10
dual-coerce, 11
dual-combine, 12
dual-is-numeric, 12
dual-log, 13
dual-math, 14
dual-math2, 15
dual-maxmin, 15
dual-show, 16
dual-summary, 17
dual_constant, *3*, 17
dual_constant_n, 18
dual_variable, *3*, 19
dual_variable_n, *3*, *6*, 19
dual_vector, *3*, 20
dual_vector-access, 20
dual_vector-class, 21
dualr-class, 21

erf, 22

erf,dualr-method *(erf)*, 22
erf,numeric-method *(erf)*, 22
erfc, 22
erfc,dualr-method *(erfc)*, 22
erfc,numeric-method *(erfc)*, 22
exp,dualr-method *(dual-math)*, 14

gradient, *3*, 23

hessian, *3*, 24

is.numeric,dualr-method
(dual-is-numeric), 12
is_dual, 24

jacobian, *3*, 25

lbeta, 26
lbeta,dualr,dualr-method *(lbeta)*, 26
lbeta,dualr,numeric-method *(lbeta)*, 26
lbeta,numeric,dualr-method *(lbeta)*, 26
lbeta,numeric,numeric-method *(lbeta)*, 26
length,dual_vector-method
(dual_vector-access), 20
log,dualr-method *(dual-log)*, 13

Math,dualr-method *(dual-math)*, 14
Math2,dualr-method *(dual-math2)*, 15
max,dualr-method *(dual-maxmin)*, 15
min,dualr-method *(dual-maxmin)*, 15

nabla *(nabla-package)*, 3
nabla-package, 3

Ops,dualr,dualr-method
(dual-arithmetic), 8
Ops,dualr,numeric-method
(dual-arithmetic), 8
Ops,numeric,dualr-method
(dual-arithmetic), 8

psigamma, 27
psigamma,dualr-method *(psigamma)*, 27
psigamma,numeric-method *(psigamma)*, 27

show,dual_vector-method *(dual-show)*, 16
show,dualr-method *(dual-show)*, 16
sqrt,dualr-method *(dual-math)*, 14
Summary,dualr-method *(dual-summary)*, 17

value, *3*, 27
value,dualr-method *(value)*, 27
value,numeric-method *(value)*, 27