

Package: weyl (via r-universe)

September 17, 2024

Type Package

Title The Weyl Algebra

Version 0.0-6

Depends methods, R (>= 3.5.0)

Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>

Description A suite of routines for Weyl algebras. Notation follows Coutinho (1995, ISBN 0-521-55119-6, ``A Primer of Algebraic D-Modules"). Uses 'disordR' discipline (Hankin 2022 <doi:10.48550/arXiv.2210.03856>). To cite the package in publications, use Hankin 2022 <doi:10.48550/arXiv.2212.09230>.

License GPL (>= 2)

LazyData yes

Suggests knitr,rmarkdown,testthat,covr

VignetteBuilder knitr

Imports disordR (>= 0.0-8), freealg (>= 1.0-4), spray (>= 1.0-19)

URL <https://github.com/RobinHankin/weyl>,
<https://robinhankin.github.io/weyl/>

BugReports <https://github.com/RobinHankin/weyl/issues>

Repository <https://robinhankin.r-universe.dev>

RemoteUrl <https://github.com/robinhankin/weyl>

RemoteRef HEAD

RemoteSha 6e0aa78a63f517d7de36c0208953da320e8cf618

Contents

weyl-package	2
coeffs	3
constant	4
degree	5

derivation	6
dim	6
dot-class	7
drop	8
grade	9
horner	10
identity	11
oom	12
Ops	12
print.weyl	14
rweyl	15
spray	16
weyl	17
weyl-class	18
x_and_d	18
zero	19

Index	20
--------------	-----------

weyl-package	<i>The Weyl Algebra</i>
--------------	-------------------------

Description

A suite of routines for Weyl algebras. Notation follows Coutinho (1995, ISBN 0-521-55119-6, "A Primer of Algebraic D-Modules"). Uses 'disordR' discipline (Hankin 2022 <doi:10.48550/arXiv.2210.03856>). To cite the package in publications, use Hankin 2022 <doi:10.48550/arXiv.2212.09230>.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Robin K. S. Hankin [aut, cre] (<<https://orcid.org/0000-0001-5982-0415>>)

Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

Examples

```
x <- rweyl(d=1)
y <- rweyl(d=1)
z <- rweyl(d=1)

is.zero(x*(y*z) - (x*y)*z) # should be TRUE
```

`coeffs`*Manipulate the coefficients of a weyl object*

Description

Manipulate the coefficients of a weyl object. The coefficients are disord objects.

Usage

```
coeffs(S) <- value
```

Arguments

<code>S</code>	A weyl object
<code>value</code>	Numeric

Details

To access coefficients of a weyl object `S`, use `spray::coeffs(S)` [package idiom is `coeffs(S)`]. Similarly to access the index matrix use `index(s)`.

The replacement method is package-specific; use `coeffs(S) <-value`.

Value

Extraction methods return a disord object (possibly dropped); replacement methods return a weyl object.

Author(s)

Robin K. S. Hankin

Examples

```
(a <- rweyl(9))
coeffs(a)
coeffs(a)[coeffs(a)<3] <- 100
a
```

`constant`*The constant term*

Description

The *constant* of a `weyl` object is the coefficient of the term with all zeros.

Usage

```
constant(x, drop = TRUE)
constant(x) <- value
```

Arguments

<code>x</code>	Object of class <code>weyl</code>
<code>drop</code>	Boolean with default <code>TRUE</code> meaning to return the value of the coefficient, and <code>FALSE</code> meaning to return the corresponding <code>Weyl</code> object
<code>value</code>	Constant value to replace existing one

Value

Returns a numeric or `weyl` object

Note

The `constant.weyl()` function is somewhat awkward because it has to deal with the difficult case where the constant is zero and `drop=FALSE`.

Author(s)

Robin K. S. Hankin

Examples

```
(a <- rweyl()+700)
constant(a)
constant(a, drop=FALSE)

constant(a) <- 0
constant(a)
constant(a, drop=FALSE)

constant(a+66) == constant(a) + 66
```

 degree

The degree of a weyl object

Description

The *degree* of a monomial weyl object $x^a \partial^b$ is defined as $a + b$. The degree of a general weyl object expressed as a linear combination of monomials is the maximum of the degrees of these monomials. Following Coutinho we have:

- $\deg(d_1 + d_2) \leq \max(\deg(d_1) + \deg(d_2))$
- $\deg(d_1 d_2) = \deg(d_1) + \deg(d_2)$
- $\deg(d_1 d_2 - d_2 d_1) \leq \deg(d_1) + \deg(d_2) - 2$

Usage
`deg(S)`
Arguments

S Object of class weyl

Value

Nonnegative integer (or $-\infty$ for the zero Weyl object)

Note

The degree of the zero object is conventionally $-\infty$.

Author(s)

Robin K. S. Hankin

Examples

```
(a <- rweyl())
deg(a)

d1 <- rweyl(n=2)
d2 <- rweyl(n=2)

deg(d1+d2) <= deg(d1) + deg(d2)
deg(d1*d2) == deg(d1) + deg(d2)
deg(d1*d2-d2*d1) <= deg(d1) + deg(d2) -2
```

 derivation

Derivations

Description

A *derivation* D of an algebra A is a linear operator that satisfies $D(d_1 d_2) = d_1 D(d_2) + D(d_1) d_2$, for every $d_1, d_2 \in A$. If a derivation is of the form $D(d) = [d, f] = df - fd$ for some fixed $f \in A$, we say that D is an *inner* derivation.

Function `as.der()` returns a derivation with `as.der(f)(g)=fg-gf`.

Usage

`as.der(S)`

Arguments

`S` Weyl object

Value

Returns a function, a derivation

Author(s)

Robin K. S. Hankin

Examples

```
(o <- rweyl(n=2,d=2))
(f <- as.der(o))

d1 <-rweyl(n=1,d=2)
d2 <-rweyl(n=2,d=2)

f(d1*d2) == d1*f(d2) + f(d1)*d2 # should be TRUE
```

 dim

The dimension of a weyl object

Description

The *dimension* of a weyl algebra is the number of variables needed; it is half the `spray::arity()`. The *dimension* of a Weyl algebra generated by $\{x_1, x_2, \dots, x_n, \partial_{x_1}, \partial_{x_2}, \dots, \partial_{x_n}\}$ is n (not $2n$).

Usage

```
## S3 method for class 'weyl'
dim(x)
```

Arguments

x Object of class weyl

Value

Integer

Note

Empty spray objects give zero-dimensional weyl objects.

Author(s)

Robin K. S. Hankin

Examples

```
(a <- rweyl())
dim(a)
```

dot-class

Class "dot"

Description

The dot object is defined so that idiom like `.[x, y]` returns the commutator, that is, $xy - yx$.

The dot object is generated by running script `inst/dot.Rmd`, which includes some further discussion and technical documentation, and creates file `dot.rda` which resides in the `data/` directory.

The `borcherds` vignette discusses this in a more general context.

Arguments

x Object of any class
i, j elements to commute
... Further arguments to `dot_error()`, currently ignored

Value

Always returns an object of the same class as `xy`.

Author(s)

Robin K. S. Hankin

Examples

```
x <- rweyl(n=1,d=2)
y <- rweyl(n=1,d=2)
z <- rweyl(n=1,d=2)

.[x,.[y,z]] + .[y,.[z,x]] + .[z,.[x,y]] # Jacobi identity
```

drop

Drop redundant information

Description

Coerce constant weyl objects to numeric

Usage

```
drop(x)
```

Arguments

x Weyl object

Details

If its argument is a constant weyl object, coerce to numeric.

Value

Returns either a length-one numeric vector or its argument, a weyl object

Note

Many functions in the package take drop as an argument which, if TRUE, means that the function returns a dropped value.

Author(s)

Robin K. S. Hankin

Examples

```
a <- rweyl() + 67
drop(a)

drop(idweyl(9))

drop(constant(a, drop=FALSE))
```

grade

The grade of a weyl object

Description

The *grade* of a homogeneous term of a Weyl algebra is the sum of the powers. Thus the grade of $4xy^2\partial_x^3\partial_y^4$ is $1 + 2 + 3 + 4 = 10$.

The functionality documented here closely follows the equivalent in the **clifford** package.

Coutinho calls this the *symbol map*.

Usage

```
grade(C, n, drop=TRUE)
grade(C,n) <- value
grades(x)
```

Arguments

C, x	Weyl object
n	Integer vector specifying grades to extract
value	Replacement value, a numeric vector
drop	Boolean, with default TRUE meaning to coerce a constant operator to numeric, and FALSE meaning not to

Details

Function `grades()` returns an (unordered) vector specifying the grades of the constituent terms. Function `grades<-()` allows idiom such as `grade(x, 1:2) <- 7` to operate as expected [here to set all coefficients of terms with grades 1 or 2 to value 7].

Function `grade(C, n)` returns a Weyl object with just the elements of grade `g`, where `g %in% n`.

The zero grade term, `grade(C, 0)`, is given more naturally by `constant(C)`.

Value

Integer vector or weyl object

Author(s)

Robin K. S. Hankin

Examples

```
a <- rweyl(30)

grades(a)
grade(a, 1:4)
grade(a, 5:9) <- -99
a
```

 horner

Horner's method

Description

Horner's method

Usage

horner(W, v)

Arguments

W	Weyl object
v	Numeric vector of coefficients

Details

Given a formal polynomial

$$p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

it is possible to express $p(x)$ in the algebraically equivalent form

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots))$$

which is much more efficient for evaluation, as it requires only n multiplications and n additions, and this is optimal.

Author(s)

Robin K. S. Hankin

See Also[oom](#)

Examples

```

horner(x,1:5)
horner(x+d,1:3)

2+x+d |> horner(1:3) |> horner(1:2)

```

identity *The identity operator*

Description

The identity operator maps any function to itself.

Usage

```

idweyl(d)
## S3 method for class 'weyl'
as.id(S)
is.id(S)

```

Arguments

d	Integer specifying dimensionality of the weyl object (twice the spray arity)
S	A weyl object

Value

A weyl object corresponding to the identity operator

Note

The identity function cannot be called “id()” because then R would not know whether to create a spray or a weyl object.

Examples

```

idweyl(7)

a <- rweyl(d=5)
a
is.id(a)
is.id(1+a-a)
as.id(a)

a == a*1
a == a*as.id(a)

```

 ooom

One over one minus

Description

Uses Taylor's theorem to give one over one minus a Weyl object

Usage

```
oom(W, n)
```

Arguments

W	Weyl object
n	Order of expansion

Author(s)

Robin K. S. Hankin

See Also

[horner](#)

Examples

```
oom(x+d, 4)

W <- x+x*d
oom(W, 4)*(1-W) == 1-W^5
```

 Ops

Arithmetic Ops Group Methods for the Weyl algebra

Description

Allows arithmetic operators such as addition, multiplication, division, integer powers, etc. to be used for weyl objects. Idiom such as $x^2 + y*z/5$ should work as expected. Addition and subtraction, and scalar multiplication, are the same as those of the **spray** package; but “*” is interpreted as functional composition, and “^” is interpreted as repeated composition. A number of helper functions are documented here (which are not designed for the end-user).

Usage

```
## S3 method for class 'weyl'
Ops(e1, e2 = NULL)
weyl_prod_helper1(a,b,c,d)
weyl_prod_helper2(a,b,c,d)
weyl_prod_helper3(a,b,c,d)
weyl_prod_univariate_onerow(S1,S2,func)
weyl_prod_univariate_nrow(S1,S2)
weyl_prod_multivariate_onerow_singlecolumn(S1,S2,column)
weyl_prod_multivariate_onerow_allcolumns(S1,S2)
weyl_prod_multivariate_nrow_allcolumns(S1,S2)
weyl_power_scalar(S,n)
```

Arguments

S, S1, S2, e1, e2	Objects of class weyl, elements of a Weyl algebra
a, b, c, d	Integers, see details
column	column to be multiplied
n	Integer power (non-negative)
func	Function used for products

Details

All arithmetic is as for spray objects, apart from $*$ and $^$. Here, $*$ is interpreted as operator concatenation: Thus, if w_1 and w_2 are Weyl objects, then w_1w_2 is defined as the operator that takes f to $w_1(w_2f)$.

Functions such as `weyl_prod_multivariate_nrow_allcolumns()` are low-level helper functions with self-explanatory names. In this context, “univariate” means the first Weyl algebra, generated by $\{x, \partial\}$, subject to $x\partial - \partial x = 1$; and “multivariate” means the algebra generated by $\{x_1, x_2, \dots, x_n, \partial_{x_1}, \partial_{x_2}, \dots, \partial_{x_n}\}$ where $n > 1$.

The product is somewhat user-customisable via option `prodfunc`, which affects function `weyl_prod_univariate_onerow()`. Currently the package offers three examples: `weyl_prod_helper1()`, `weyl_prod_helper2()`, and `weyl_prod_helper3()`. These are algebraically identical but occupy different positions on the efficiency-readability scale. The option defaults to `weyl_prod_helper3()`, which is the fastest but most opaque. The vignette provides further details, motivation, and examples.

Powers, as in x^n , are defined in the usual way. Negative powers will always return an error.

Value

Generally, return a weyl object

Note

Function `weyl_prod_univariate_nrow()` is present for completeness, it is not used in the package

Author(s)

Robin K. S. Hankin

Examples

```
x <- rweyl(n=1,d=2)
y <- rweyl(n=1,d=2)
z <- rweyl(n=2,d=2)

x*(y+z) == x*y + x*z
is.zero(x*(y*z) - (x*y)*z)
```

```
print.weyl          Print methods for weyl objects
```

Description

Printing methods for weyl objects follow those for the **spray** package, with some additional functionality.

Usage

```
## S3 method for class 'weyl'
print(x, ...)
```

Arguments

```
x          A weyl object
...        Further arguments, currently ignored
```

Details

Option `polyform` determines whether the object is to be printed in matrix form or polynomial form: as in the **spray** package, this option governs dispatch to either `print_spray_polyform()` or `print_spray_matrixform()`.

```
> a <- rweyl()
> a      # default print method
A member of the Weyl algebra:
  x  y  z dx dy dz   val
  1  2  2  2  1  0 =   3
  2  2  0  0  1  1 =   2
  0  0  0  1  1  2 =   1
> options(polyform = TRUE)
> a
A member of the Weyl algebra:
+3*x*y^2*z^2*dx^2*dy +2*x^2*y^2*dy*dz +dx*dy*dz^2
> options(polyform = FALSE) # restore default
```

Irrespective of the value of `polyform`, option `weylvars` controls the variable names. If `NULL` (the default), then sensible values are used: either `[xyz]` if the dimension is three or less, or integers. But option `weylvars` is user-settable:

```
> options(weylvars=letters[18:20])
> a
A member of the Weyl algebra:
  r  s  t dr ds dt   val
  1  2  2  2  1  0 =   3
  2  2  0  0  1  1 =   2
  0  0  0  1  1  2 =   1
> options(polyform=TRUE)
> a
A member of the Weyl algebra:
+3*r*s^2*t^2*dr^2*ds +2*r^2*s^2*ds*dt +dr*ds*dt^2
> options(polyform=FALSE) ; options(weylvars=NULL)
```

If the user sets `weylvars`, the print method tries to do the Right Thing (tm). If set to `c("a", "b", "c")`, for example, the generators are named `c(" a", " b", " c", "da", "db", "dc")` [note the spaces]. If the algebra is univariate, the names will be something like `d` and `x`. No checking is performed and if the length is not equal to the dimension, undesirable behaviour may occur. For the love of God, do not use a variable named `d`. Internally, `weylvars` works by changing the `sprayvars` option in the **spray** package.

Note that, as for `spray` objects, this option has no algebraic significance: it only affects the print method.

Value

Returns a `weyl` object.

Author(s)

Robin K. S. Hankin

Examples

```
a <- rweyl()
print(a)
options(polyform=TRUE)
print(a)
```

rweyl

Random weyl objects

Description

Creates random weyl objects: quick-and-dirty examples of Weyl algebra elements

Usage

```
rweyl(nterms = 3, vals = seq_len(nterms), dim = 3, powers = 0:2)
```

Arguments

nterms	Number of terms in output
vals	Values of coefficients
dim	Dimension of weyl object
powers	Set from which to sample the entries of the index matrix

Value

Returns a weyl object

Author(s)

Robin K. S. Hankin

Examples

```
rweyl()
rweyl(d=7)
```

spray

Create spray objects

Description

Function `spray()` creates a sparse array; function `weyl()` coerces a spray object to a Weyl object.

Usage

```
spray(M, x, addrepeats=FALSE)
```

Arguments

M	An integer-valued matrix, the index of the weyl object
x	Numeric vector of coefficients
addrepeats	Boolean, specifying whether repeated rows are to be added

Details

The function is discussed and motivated in the **spray** package.

Value

Return a weyl or a Boolean

Author(s)

Robin K. S. Hankin

Examples

```
(W <- spray(matrix(1:36,6,6),1:6))
weyl(W)

as.weyl(15,d=3)
```

weyl

The algebra and weyl objects

Description

Basic functions for weyl objects

Usage

```
weyl(M)
is.weyl(M)
as.weyl(val,d)
is.ok.weyl(M)
```

Arguments

M	A weyl or spray object
val, d	Value and dimension for weyl object

Details

To create a weyl object, pass a spray to function `weyl()`, as in `weyl(M)`. To create a spray object to pass to `weyl()`, use function `spray()`, which is a synonym for `spray : spray()`.

Function `weyl()` is the formal creator method; `is.weyl()` tests for weyl objects and `is.ok.weyl()` checks for well-formed sprays. Function `as.weyl()` tries (but not very hard) to infer what the user intended and return the right thing.

Value

Return a weyl or a Boolean

Author(s)

Robin K. S. Hankin

Examples

```
(W <- spray(matrix(1:36,6,6),1:6))
weyl(W)

as.weyl(15,d=3)

is.ok.weyl(spray(matrix(1:30,5,6)))
is.ok.weyl(spray(matrix(1:30,6,5)))
```

weyl-class

Class "weyl"

Description

The formal S4 class for weyls.

Objects from the Class

Objects *can* be created by calls of the form `new("weyl", ...)` but this is not encouraged. Use functions `weyl()` or `as.weyl()` instead.

Author(s)

Robin K. S. Hankin

x_and_d

Generating elements for the first Weyl algebra

Description

Variables `x` and `d` correspond to operator x and ∂_x ; they are provided for convenience. These elements generate the one-dimensional Weyl algebra.

Note that a similar system for multivariate Weyl algebras is not desirable. We might want to consider the Weyl algebra generated by $\{x, y, z, \partial_x, \partial_y, \partial_z\}$ and correspondingly define \mathbb{R} variables x, y, z, dx, dy, dz . But then variable `x` is ambiguous: is it a member of the first Weyl algebra or the third?

Usage

```
data(x_and_d)
```

Author(s)

Robin K. S. Hankin

Examples

```

d
x

.[d,x] # dx-xd==1

d^3 * x^4

(1-d*x*d)*(x^2-d^3)

```

zero

The zero operator

Description

The zero operator maps any function to the zero function (which maps anything to zero). To test for being zero, use `spray::is.zero()`; package idiom would be `is.zero()`.

Usage

```
zero(d)
```

Arguments

`d` Integer specifying dimensionality of the weyl object (twice the spray arity)

Value

A weyl object corresponding to the zero operator (or a Boolean for `is.zero()`)

Examples

```

(a <- rweyl(d=5))
is.zero(a)
is.zero(a-a)
is.zero(a*0)

a == a + zero(dim(a))

```

Index

- * **classes**
 - weyl-class, 18
- * **datasets**
 - x_and_d, 18
- * **package**
 - weyl-package, 2
- * **symbolmath**
 - horner, 10
- . (dot-class), 7
- [,dot,ANY,ANY-method (dot-class), 7
- [,dot,ANY,missing-method (dot-class), 7
- [,dot,matrix,matrix-method (dot-class), 7
- [,dot,missing,ANY-method (dot-class), 7
- [,dot,missing,missing-method (dot-class), 7
- [,dot-method (dot-class), 7
- [.dot (dot-class), 7

- as.der (derivation), 6
- as.id (identity), 11
- as.identity (identity), 11
- as.one.weyl (identity), 11
- as.weyl (weyl), 17

- coeff (coeffs), 3
- coeffs, 3
- coeffs,weyl-method (coeffs), 3
- coeffs.weyl (coeffs), 3
- coeffs<- (coeffs), 3
- coeffs<-,weyl-method (coeffs), 3
- coeffs<- .weyl (coeffs), 3
- commutator (dot-class), 7
- const (constant), 4
- constant, 4
- constant<- (constant), 4

- d (x_and_d), 18
- deg (degree), 5
- degree, 5

- derivation, 6
- derivations (derivation), 6
- dim, 6
- dimension (dim), 6
- dot (dot-class), 7
- dot-class, 7
- dot_error (dot-class), 7
- drop, 8
- drop,weyl-method (drop), 8

- empty (zero), 19
- extract (dot-class), 7

- grade, 9
- grade<- (grade), 9
- grades (grade), 9

- horner, 10, 12

- id (identity), 11
- identity, 11
- idweyl (identity), 11
- index (coeffs), 3
- is.empty (zero), 19
- is.id (identity), 11
- is.identity (identity), 11
- is.ok.weyl (weyl), 17
- is.weyl (weyl), 17
- is.zero (zero), 19

- jacobi (dot-class), 7

- oom, 10, 12
- Ops, 12

- print (print.weyl), 14
- print.weyl, 14
- prodfunc (Ops), 12

- rweyl, 15
- spray, 16

symbol_map (grade), 9

value (coeffs), 3
value, weyl-method (coeffs), 3
value.weyl (coeffs), 3
value<- (coeffs), 3
values (coeffs), 3

weyl, 17
weyl-class, 18
weyl-package, 2
weyl_power_scalar (Ops), 12
weyl_prod (Ops), 12
weyl_prod_helper1 (Ops), 12
weyl_prod_helper2 (Ops), 12
weyl_prod_helper3 (Ops), 12
weyl_prod_multivariate_nrow_allcolumns
 (Ops), 12
weyl_prod_multivariate_onerow_allcolumns
 (Ops), 12
weyl_prod_multivariate_onerow_singlecolumn
 (Ops), 12
weyl_prod_univariate_nrow (Ops), 12
weyl_prod_univariate_onerow (Ops), 12
weylvars (print.weyl), 14

x (x_and_d), 18
x_and_d, 18

zero, 19