Package ‘stokes’

June 4, 2024

Type Package

Title The Exterior Calculus

Version 1.2-1

Depends R (>= 3.5.0)

Suggests knitr, Deriv, testthat, markdown, rmarkdown, quadform, magrittr, covr

VignetteBuilder knitr

Imports permutations (>= 1.1-2), partitions, methods, disordR (>= 0.9-7), spray (>= 1.0-25)

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


License GPL-2

LazyData yes

URL https://github.com/RobinHankin/stokes

BugReports https://github.com/RobinHankin/stokes/issues

NeedsCompilation no

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

Repository CRAN

Date/Publication 2024-06-04 15:30:06 UTC
Contents

stokes-package ......................................................... 2
Alt ................................................................. 5
as.1form .......................................................... 7
coeffs ............................................................ 8
consolidate ....................................................... 9
contract .......................................................... 10
dovs ............................................................. 11
dx ............................................................... 12
ex ................................................................. 13
hodge ............................................................. 14
inner ............................................................... 15
issmall .......................................................... 17
keep .............................................................. 17
kform ............................................................. 18
kinner ............................................................ 21
ktensor ........................................................... 22
Ops.kform ....................................................... 23
print.stokes ...................................................... 25
rform ............................................................. 27
scalar ............................................................ 28
summary.stokes ................................................... 29
symbolic ........................................................ 30
tensorprod ...................................................... 32
transform ........................................................ 33
vector_cross_product ........................................... 35
volume .......................................................... 36
wedge ............................................................ 38
zap ................................................................. 39
zero ............................................................... 40

Index .............................................................. 42

Description


stokes-package The Exterior Calculus
**Details**

The DESCRIPTION file:

Package: stokes
Type: Package
Title: The Exterior Calculus
Version: 1.2-1
Depends: R (>= 3.5.0)
Suggests: knitr, Deriv, testthat, markdown, rmarkdown, quadform, magrittr, covr
VignetteBuilder: knitr
Imports: permutations (>= 1.1-2), partitions, methods, disordR (>= 0.9-7), spray (>= 1.0-25)
Authors@R: person(given=c("Robin", "K. S."), family="Hankin", role = c("aut","cre"), email="hankin.robin@gmail.com")
Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>
Description: Provides functionality for working with tensors, alternating forms, wedge products, Stokes’s theorem, and
License: GPL-2
LazyData: yes
URL: https://github.com/RobinHankin/stokes
BugReports: https://github.com/RobinHankin/stokes/issues
Author: Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Index of help topics:

Alt Alternating multilinear forms
Ops.kform Arithmetic Ops Group Methods for 'kform' and 'ktensor' objects
as.1form Coerce vectors to 1-forms
coeffs Extract and manipulate coefficients
consolidate Various low-level helper functions
contract Contractions of k-forms
dovs Dimension of the underlying vector space
dx Elementary forms in three-dimensional space
ex Basis vectors in three-dimensional space
hodge Hodge star operator
inner Inner product operator
issmall Is a form zero to within numerical precision?
keep Keep or drop variables
kform k-forms
kinner Inner product of two kforms
ktensor k-tensors
print.stokes Print methods for k-tensors and k-forms
rform Random kforms and ktensors
scalar Scalars and losing attributes
stokes-package The Exterior Calculus
summary.stokes Summaries of tensors and alternating forms
symbolic Symbolic form
tensorprod Tensor products of k-tensors
transform Linear transforms of k-forms
vector_cross_product  The Vector cross product
volume       The volume element
wedge        Wedge products
zap          Zap small values in k-forms and k-tensors
zero         Zero tensors and zero forms

Generally in the package, arguments that are $k$-forms are denoted $K$, $k$-tensors by $U$, and spray objects by $S$. Multilinear maps (which may be either $k$-forms or $k$-tensors) are denoted by $M$.

Author(s)
Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)
Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

References

See Also
spray

Examples
```r
## Some k-tensors:
U1 <- as.ktensor(matrix(1:15,5,3))
U2 <- as.ktensor(cbind(1:3,2:4),1:3)

## Coerce a tensor to functional form, here mapping $V^3 \rightarrow \mathbb{R}$ (here $V=\mathbb{R}^{15}$):
as.function(U1)(matrix(rnorm(45),15,3))

## Tensor product is tensorprod() or %X%:
U1 %X% U2

## A k-form is an alternating k-tensor:
K1 <- as.kform(cbind(1:5,2:6),rnorm(5))
K2 <- kform_general(3:6,2,1:6)
K3 <- rform(9,3,9,runif(9))

## The distributive law is true
(K1 + K2) ^ K3 == K1 ^ K3 + K2 ^ K3 # TRUE to numerical precision

## Wedge product is associative (non-trivial):
(K1 ^ K2) ^ K3
```
\[ K1 \wedge (K2 \wedge K3) \]

## k-forms can be coerced to a function and wedge product:
\[
f <- \text{as.function}(K1 \wedge K2 \wedge K3)\]

## \(E\) is a random point in \(V^k\):
\[
E <- \text{matrix(rnorm(63),9,7)}\]

## \(f()\) is alternating:
\[
f(E)
f(E[,7:1])\]

## The package blurs the distinction between symbolic and numeric computing:
\[
dx <- \text{as.kform}(1)
dy <- \text{as.kform}(2)
dz <- \text{as.kform}(3)
dx \wedge dy \wedge dz\]
\[
K3 \wedge dx \wedge dy \wedge dz\]

---

**Alt**

**Alternating multilinear forms**

**Description**

Converts a \(k\)-tensor to alternating form

**Usage**

\[
\text{Alt}(S, \text{give.kform})\]

**Arguments**

- **S**: A multilinear form, an object of class ktensor
- **give.kform**: Boolean, with default FALSE meaning to return an alternating \(k\)-tensor [that is, an object of class ktensor that happens to be alternating] and TRUE meaning to return a \(k\)-form [that is, an object of class kform]

**Details**

Given a \(k\)-tensor \(T\), we have
\[
\text{Alt}(T)(v_1, \ldots, v_k) = \frac{1}{k!} \sum_{\sigma \in S_k} \text{sgn}(\sigma) \cdot T(v_{\sigma(1)}, \ldots, v_{\sigma(k)})
\]
Thus for example if $k = 3$:

$$\text{Alt}(T)(v_1, v_2, v_3) = \frac{1}{6} \begin{pmatrix} +T(v_1, v_2, v_3) & -T(v_1, v_3, v_2) \\ -T(v_2, v_1, v_3) & +T(v_2, v_3, v_1) \\ +T(v_3, v_1, v_2) & -T(v_3, v_2, v_1) \end{pmatrix}$$

and it is reasonably easy to see that $\text{Alt}(T)$ is alternating, in the sense that

$$\text{Alt}(T)(v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k) = -\text{Alt}(T)(v_1, \ldots, v_j, \ldots, v_i, \ldots, v_k)$$

Function $\text{Alt}()$ is intended to take and return an object of class $\text{ktensor}$; but if given a $\text{kform}$ object, it just returns its argument unchanged.

A short vignette is provided with the package: type vignette("Alt") at the commandline.

**Value**

Returns an alternating $k$-tensor. To work with $k$-forms, which are a much more efficient representation of alternating tensors, use $\text{as.kform}()$.

**Author(s)**

Robin K. S. Hankin

**See Also**

$kform$

**Examples**

```r
(X <- ktensor(spray(rbind(1:3),6)))
Alt(X)
Alt(X,give_kform=TRUE)

S <- as.ktensor(expand.grid(1:3,1:3),rnorm(9))
S
Alt(S)

issmall(Alt(S) - Alt(Alt(S))) # should be TRUE; Alt() is idempotent

a <- rtensor()
V <- matrix(rnorm(21),ncol=3)
LHS <- as.function(Alt(a))(V)
RHS <- as.function(Alt(a,give_kform=TRUE))(V)
c(LHS=LHS,RHS=RHS,diff=LHS-RHS)
```
as.1form

Coerce vectors to 1-forms

Description
Given a vector, return the corresponding 1-form; the exterior derivative of a 0-form (that is, a scalar function). Function grad() is a synonym.

Usage
as.1form(v)
grad(v)

Arguments
v A vector with element $i$ being $\partial f/\partial x_i$

Details
The exterior derivative of a $k$-form $\phi$ is a $(k+1)$-form $d\phi$ given by

$$d\phi(P_x(v_i, \ldots, v_{k+1})) = \lim_{h \to 0} \frac{1}{h^{k+1}} \int_{\partial P_x(hv_1, \ldots, hv_{k+1})} \phi$$

We can use the facts that

$$d(f \, dx_{i_1} \wedge \cdots \wedge dx_{i_k}) = df \wedge dx_{i_1} \wedge \cdots \wedge dx_{i_k}$$

and

$$df = \sum_{j=1}^{n} (D_j f) \, dx_j$$

to calculate differentials of general $k$-forms. Specifically, if

$$\phi = \sum_{1 \leq i_1 < \cdots < i_k \leq n} a_{i_1 \ldots i_k} \, dx_{i_1} \wedge \cdots \wedge dx_{i_k}$$

then

$$d\phi = \sum_{1 \leq i_1 < \cdots < i_k \leq n} \left[ \sum_{j=1}^{n} D_j a_{i_1 \ldots i_k} \, dx_j \right] \wedge dx_{i_1} \wedge \cdots \wedge dx_{i_k}.$$ 

The entry in square brackets is given by grad(). See the examples for appropriate R idiom.

Value
A one-form


Author(s)

Robin K. S. Hankin

See Also

kform

Examples

as.1form(1:9)  # note ordering of terms

as.1form(rnorm(20))

grad(c(4,7)) ^ grad(1:4)

---

coeffs  

Extract and manipulate coefficients

Description

Extract and manipulate coefficients of ktensor and kform objects; this using the methods of the spray package.

Functions as spray() and nterms() are imported from spray.

Details

To see the coefficients of a kform or ktensor object, use coeffs(), which returns a disord object (this is actually spray::coeffs()). Replacement methods also use the methods of the spray package. Note that disorder discipline is enforced.

Experimental functionality for “pure” extraction and replacement is provided, following spray version 1.0-25 or above. Thus idiom such as a[abs(coeffs(a)) > 0.1] or indeed a[coeffs(a) < 1] <- 0 should work as expected.

Author(s)

Robin K. S. Hankin
consolidate

Examples
(a <- kform_general(5,2,1:10))
coeffs(a) # a disord object
coeffs(a)[coeffs(a)%%2==1] <- 100 # replace every odd coeff with 100
a
coeffs(a*0)
a <- rform()
a[coeffs(a) < 5] # experimental
a[coeffs(a) > 3] <- 99 # experimental

consolidate Various low-level helper functions

Description
Various low-level helper functions used in Alt() and kform()

Usage
consolidate(S)
kill_trivial_rows(S)
include_perms(S)
ktensor_to_kform(S)

Arguments
S Object of class spray

Details
Low-level helper functions.

- Function consolidate() takes a spray object, and combines any rows that are identical up to a permutation, respecting the sign of the permutation
- Function kill_trivial_rows() takes a spray object and deletes any rows with a repeated entry (which have k-forms identically zero)
- Function include_perms() replaces each row of a spray object with all its permutations, respecting the sign of the permutation
- Function ktensor_to_kform() coerces a k-form to a k-tensor

Value
The functions documented here all return a spray object.
合同

Author(s)
Robin K. S. Hankin

See Also
ktensor, kform, Alt

Examples
(S <- spray(matrix(c(1,1,2,2,1,3,3,1,3,5),ncol=2,byrow=TRUE),1:5))
kill_trivial_rows(S)  # (rows 1 and 3 killed, repeated entries)
consolidate(S)  # (merges rows 2 and 4)
include_perms(S)  # returns a spray object, not alternating tensor.

Description
Contractions of k-forms

A contraction is a natural linear map from k-forms to k − 1-forms.

Usage
contract(K, v, lose=TRUE)
contract_elementary(o, v)

Arguments
K  A k-form
o  Integer-valued vector corresponding to one row of an index matrix
lose  Boolean, with default TRUE meaning to coerce a 0-form to a scalar and FALSE meaning to return the formal 0-form
v  A vector; in function contract(), if a matrix, interpret each column as a vector to contract with

Details
Given a k-form φ and a vector v, the contraction φ_v of φ and v is a k − 1-form with

\[ φ_v(v^1, \ldots, v^{k-1}) = φ(v, v^1, \ldots, v^{k-1}) \]

provided k > 1; if k = 1 we specify φ_v = φ(v).

Function contract_elementary() is a low-level helper function that translates elementary k-forms with coefficient 1 (in the form of an integer vector corresponding to one row of an index matrix) into its contraction with v.

There is an extensive vignette in the package, vignette("contract").
Value

Returns an object of class kform.

Author(s)

Robin K. S. Hankin

References


See Also

wedge, lose

Examples

contract(as.kform(1:5),1:8)
contract(as.kform(1),3)  # 0-form

contract_elementary(c(1,2,5),c(1,2,10,11,71))

## Now some verification [takes ~10s to run]:
#o <- kform(spray(t(replicate(2, sample(9,4))), runif(2)))
#V <- matrix(rnorm(36),ncol=4)
#jj <- c(
#  as.function(o)(V),
#  as.function(contract(o,V[,1,drop=TRUE]))(V[,-1]), # scalar
#  as.function(contract(o,V[,1:2]))(V[,-(1:2),drop=FALSE]),
#  as.function(contract(o,V[,1:3]))(V[,-(1:3),drop=FALSE]),
#  as.function(contract(o,V[,1:4],lose=FALSE))(V[,-(1:4),drop=FALSE])
#)

#print(jj)
#max(jj) - min(jj) # zero to numerical precision
**Description**

A $k$-form $\omega \in \Lambda^k(V)$ maps $V^k$ to the reals, where $V = \mathbb{R}^n$. Function `dovs()` returns $n$, the dimensionality of the underlying vector space. The function itself is almost trivial, returning the maximum of the index matrix.

Special dispensation is given for zero-forms and zero tensors, which return zero.

Vignette `dovs` provides more discussion.

**Usage**

```r
dovs(K)
```

**Arguments**

- `K`  
  A $k$-form or $k$-tensor

**Value**

Returns a non-negative integer

**Author(s)**

Robin K. S. Hankin

**Examples**

```r
dovs(rform())
table(replicate(20, dovs(rform(3))))
```

---

## dx

*Elementary forms in three-dimensional space*

**Description**

Objects `dx`, `dy` and `dz` are the three elementary one-forms on three-dimensional space. These objects can be generated by running script `vignettes/dx.Rmd`, which includes some further discussion and technical documentation and creates file `dx.rda` which resides in the `data/` directory.

The default print method is a little opaque for these objects. To print them more intuitively, use

```r
options(kform_symbolic_print = "dx")
```

which is documented at `print.Rd`.

**Usage**

```r
data(dx)
```
Details

See vignettes dx and exeyez for an extended discussion; a use-case is given in vector_cross_product.

Author(s)

Robin K. S. Hankin

References

• M. Spivak 1971. *Calculus on manifolds*, Addison-Wesley

See Also

d, print.kform

Examples

dx
hodge(dx)
hodge(dx, 3)

dx # default print method, not particularly intelligible
options(kform_symbolic_print = 'dx')  # shows dx dy dz
dx
dx*dz
hodge(dx, 3)
as.function(dx)(ex)

options(kform_symbolic_print = NULL)  # revert to default

---

**ex**

*Basis vectors in three-dimensional space*

Description

Objects ex, ey and ez are the three elementary one-forms on three-dimensional space, sometimes denoted \((e_x, e_y, e_z)\). These objects can be generated by running script `vignettes/ex.Rmd`, which includes some further discussion and technical documentation and creates file `exeyez.rda` which resides in the `data/` directory.

Details

See vignettes dx and ex for an extended discussion; a use-case is given in vector_cross_product.
hodge

Author(s)
Robin K. S. Hankin

References
- M. Spivak 1971. *Calculus on manifolds*, Addison-Wesley

See Also
d.print.kform

Examples
```r
as.function(dx)(ex)
(X <- as.kform(matrix(1:12,nrow=4),c(1,2,7,11)))
as.function(X)(cbind(e(2,12),e(6,12),e(10,12)))
```

---

**hodge**

*Hodge star operator*

Description
Given a \(k\)-form, return its Hodge dual

Usage
```r
hodge(K, n=dovs(K), g, lose=TRUE)
```

Arguments
- \(K\) Object of class kform
- \(n\) Dimensionality of space, defaulting the the largest element of the index
- \(g\) Diagonal of the metric tensor, with missing default being the standard metric of the identity matrix. Currently, only entries of \(\pm 1\) are accepted
- \(lose\) Boolean, with default \(TRUE\) meaning to coerce to a scalar if appropriate

Value
Given a \(k\)-form, in an \(n\)-dimensional space, return a \((n - k)\)-form.

Note
Most authors write the Hodge dual of \(\psi\) as \(*\psi\) or \(\star\psi\), but Weintraub uses \(\psi^*\).
Author(s)
Robin K. S. Hankin

See Also
wedge

Examples

(o <- kform_general(5,2,1:10))
hodge(o)
o == hodge(hodge(o))

Faraday <- kform_general(4,2,runif(6)) # Faraday electromagnetic tensor
mink <- c(-1,1,1,1) # Minkowski metric
hodge(Faraday,g=mink)

Faraday == Faraday |> hodge(g=mink) |> hodge(g=mink) |> hodge(g=mink)
hodge(dx,3) == dy*dz

## Some edge-cases:
hodge(scalar(1),2)
hodge(zeroform(5),9)
hodge(volume(5))
hodge(volume(5),lose=TRUE)
hodge(scalar(7),n=9)

inner  Inner product operator

Description
The inner product

Usage
inner(M)
Arguments

M  square matrix

Details

The inner product of two vectors x and y is usually written \( \langle x, y \rangle \) or \( x \cdot y \), but the most general form would be \( x^T M y \) where \( M \) is a matrix. Noting that inner products are multilinear, that is \( \langle x, ay + bz \rangle = a \langle x, y \rangle + b \langle x, z \rangle \) and \( \langle ax + by, z \rangle = a \langle x, z \rangle + b \langle y, z \rangle \), we see that the inner product is indeed a multilinear map, that is, a tensor.

Given a square matrix \( M \), function \texttt{inner(M)} returns the 2-form that maps \( x, y \) to \( x^T M y \). Non-square matrices are effectively padded with zeros.

A short vignette is provided with the package: type \texttt{vignette("inner")} at the commandline.

Value

Returns a \( k \)-tensor, an inner product

Author(s)

Robin K. S. Hankin

See Also

\texttt{kform}

Examples

inner(diag(7))
inner(matrix(1:9,3,3))

## Compare the following two:
Alt(inner(matrix(1:9,3,3)))  # An alternating k tensor
as.kform(inner(matrix(1:9,3,3)))  # Same thing coerced to a kform

f <- as.function(inner(diag(7)))
X <- matrix(rnorm(14),ncol=2)  # random element of \((R^7)^2\)
f(X) - sum(X[,1]*X[,2])  # zero to numerical precision

## verify positive-definiteness:
g <- as.function(inner(crossprod(matrix(rnorm(56),8,7))))
stopifnot(g(kronecker(rnorm(7),t(c(1,1))))>0)
Is a form zero to within numerical precision?

Description

Given a $k$-form, return TRUE if it is “small”

Usage

issmall(M, tol=1e-8)

Arguments

M
Object of class kform or ktensor
tol
Small tolerance, defaulting to 1e-8

Value

Returns a logical

Author(s)

Robin K. S. Hankin

Examples

o <- kform_general(3,2,runif(3))
M <- matrix(rnorm(9),3,3)
discrepancy <- o - pullback(pullback(o,M),solve(M))
discrepancy # print method might imply coefficients are zeros
issmall(discrepancy) # should be TRUE
is.zero(discrepancy) # might be FALSE

Keep or drop variables

Description

Keep or drop variables

Usage

keep(K, yes)
discard(K, no)
Arguments

K Object of class kform
yes, no Specification of dimensions to either keep (yes) or discard (no)

Details

Function keep(omega, yes) keeps the terms specified and discard(omega, no) discards the terms specified. It is not clear to me what these functions mean from a mathematical perspective.

Value

The functions documented here all return a kform object.

Author(s)

Robin K. S. Hankin

See Also

lose

Examples

(o <- kform_general(7,3,seq_len(choose(7,3))))
keep(o,1:4)  # keeps only terms with dimensions 1-4
discard(o,1:2)  # loses any term with a "1" in the index

---

kform k-forms

Description

Functionality for dealing with k-forms

Usage

kform(S)
as.kform(M,coeffs,lose=TRUE)
kform_basis(n, k)
kform_general(W,k,coeffs,lose=TRUE)
is.kform(x)
d(i)  
e(i,n)
## S3 method for class 'kform'
as.function(x,...)
Arguments

- **n**: Dimension of the vector space \( V = \mathbb{R}^n \)
- **i**: Integer
- **k**: A \( k \)-form maps \( V^k \) to \( \mathbb{R} \)
- **w**: Integer vector of dimensions
- **M, coeffs**: Index matrix and coefficients for a \( k \)-form
- **S**: Object of class spray
- **lose**: Boolean, with default TRUE meaning to coerce a 0-form to a scalar and FALSE meaning to return the formal 0-form
- **x**: Object of class kform

... Further arguments, currently ignored

Details

A \( k \)-form is an alternating \( k \)-tensor. In the package, \( k \)-forms are represented as sparse arrays (spray objects), but with a class of c("kform", "spray"). The constructor function kform() takes a spray object and returns a kform object: it ensures that rows of the index matrix are strictly non-negative integers, have no repeated entries, and are strictly increasing. Function as.kform() is more user-friendly.

- kform() is the constructor function. It takes a spray object and returns a kform.
- as.kform() also returns a kform but is a bit more user-friendly than kform().
- kform_basis() is a low-level helper function that returns a matrix whose rows constitute a basis for the vector space \( \Lambda^k(\mathbb{R}^n) \) of \( k \)-forms.
- kform_general() returns a kform object with terms that span the space of alternating tensors.
- is.kform() returns TRUE if its argument is a kform object.
- d() is an easily-typed synonym for as.kform(). The idea is that d(1) = dx, d(2)=dy, d(5)=dx^5, etc. Also note that, for example, d(1:3)=dx*dy*dz, the volume form.

Recall that a \( k \)-tensor is a multilinear map from \( V^k \) to the reals, where \( V = \mathbb{R}^n \) is a vector space. A multilinear \( k \)-tensor \( T \) is alternating if it satisfies

\[
T(v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k) = -T(v_1, \ldots, v_j, \ldots, v_i, \ldots, v_k)
\]

In the package, an object of class kform is an efficient representation of an alternating tensor.

Function kform_basis() is a low-level helper function that returns a matrix whose rows constitute a basis for the vector space \( \Lambda^k(\mathbb{R}^n) \) of \( k \)-forms:

\[
\phi = \sum_{1 \leq i_1 < \cdots < i_k \leq n} a_{i_1 \ldots i_k} dx_{i_1} \wedge \cdots \wedge dx_{i_k}
\]

and indeed we have:

\[
a_{i_1 \ldots i_k} = \phi(e_{i_1}, \ldots, e_{i_k})
\]

where \( e_j, 1 \leq j \leq k \) is a basis for \( V \).
Value

All functions documented here return a kform object except as.function.kform(), which returns a function, and is.kform(), which returns a Boolean, and e(), which returns a conjugate basis to that of d().

Note

Hubbard and Hubbard use the term “k-form”, but Spivak does not.

Author(s)

Robin K. S. Hankin

References

Hubbard and Hubbard; Spivak

See Also

ktensor, lose

Examples

as.kform(cbind(1:5,2:6),rnorm(5))
kform_general(1:4,2,coeffs=1:6) # used in electromagnetism

K1 <- as.kform(cbind(1:5,2:6),rnorm(5))
K2 <- kform_general(5:8,2,1:6)
K1*K2 # or wedge(K1,K2)

d(1:3)
dx^dy^dz # same thing

d(sample(9)) # coeff is +/-1 depending on even/odd permutation of 1:9

f <- as.function(wedge(K1,K2))
E <- matrix(rnorm(32),8,4)
f(E) + f(E[,c(1,3,2,4)]) # should be zero by alternating property

options(kform_symbolic_print = 'd')
(d(5)+d(7)) ^ (d(2)^d(5) + 6*d(4)^d(7))
options(kform_symbolic_print = NULL) # revert to default
Description

Given two $k$-forms $\alpha$ and $\beta$, return the inner product $\langle \alpha, \beta \rangle$. Here our underlying vector space $V$ is $\mathbb{R}^n$.

The inner product is a symmetric bilinear form defined in two stages. First, we specify its behaviour on decomposable $k$-forms $\alpha = \alpha_1 \wedge \cdots \wedge \alpha_k$ and $\beta = \beta_1 \wedge \cdots \wedge \beta_k$ as

$$\langle \alpha, \beta \rangle = \det \left( \langle \alpha_i, \beta_j \rangle \right)_{1 \leq i,j \leq n}$$

and secondly, we extend to the whole of $\Lambda^k(V)$ through linearity.

Usage

```
kinner(o1,o2,M)
```

Arguments

- `o1, o2` Objects of class `kform`
- `M` Matrix

Value

Returns a real number

Note

There is a vignette available: type `vignette("kinner")` at the command line.

Author(s)

Robin K. S. Hankin

See Also

- `hodge`

Examples

```
a <- (2*dx)*(3*dy)
b <- (5*dx)*(7*dy)
kinner(a,b)
det(matrix(c(2*5,0,0,3*7),2,2))  # mathematically identical, slight numerical mismatch
```
**Description**

Functionality for \( k \)-tensors

**Usage**

- `ktensor(S)`
- `as.ktensor(M, coeffs)`
- `is.ktensor(x)`
- `## S3 method for class 'ktensor'`
- `as.function(x, ...)`

**Arguments**

- `M, coeffs` Matrix of indices and coefficients, as in `spray(M, coeffs)`
- `S` Object of class `spray`
- `x` Object of class `ktensor`
- `...` Further arguments, currently ignored

**Details**

A \( k \)-tensor object \( S \) is a map from \( V^k \) to the reals \( R \), where \( V \) is a vector space (here \( R^n \)) that satisfies multilinearity:

\[
S(v_1, \ldots, av_i, \ldots, v_k) = a \cdot S(v_1, \ldots, v_i, \ldots, v_k)
\]

and

\[
S(v_1, \ldots, v_i + v_i', \ldots, v_k) = S(v_1, \ldots, v_i, \ldots, x_u) + S(v_1, \ldots, v_i', \ldots, v_k).
\]

Note that this is *not* equivalent to linearity over \( V^{nk} \) (see examples).

In the `stokes` package, \( k \)-tensors are represented as sparse arrays (spray objects), but with a class of `c("ktensor", "spray")`. This is a natural and efficient representation for tensors that takes advantage of sparsity using `spray` package features.

Function `as.ktensor()` will coerce a \( k \)-form to a \( k \)-tensor via `kform_to_ktensor()`.

**Value**

All functions documented here return a `ktensor` object except `as.function.ktensor()`, which returns a function.

**Author(s)**

Robin K. S. Hankin
**Ops.kform**

**References**

Spivak 1961

**See Also**

tensorprod, kform, wedge

**Examples**

```r
as.ktensor(cbind(1:4,2:5,3:6),1:4)
```

```r
## Test multilinearity:
k <- 4
n <- 5
u <- 3

## Define a randomish k-tensor:
S <- ktensor(spray(matrix(1+sample(u*k)%%n,u,k),seq_len(u)))

## And a random point in V^k:
E <- matrix(rnorm(n*k),n,k)
E1 <- E2 <- E3 <- E
x1 <- rnorm(n)
x2 <- rnorm(n)
r1 <- rnorm(1)
r2 <- rnorm(1)

# change one column:
E1[,2] <- x1
E2[,2] <- x2
E3[,2] <- r1*x1 + r2*x2

f <- as.function(S)
r1*f(E1) + r2*f(E2) - f(E3) # should be small

## Note that multilinearity is different from linearity:
r1*f(E1) + r2*f(E2) - f(r1*E1 + r2*E2) # not small!
```

---

**Ops.kform**

Arithmetic Ops Group Methods for kform and ktensor objects

**Description**

允許算術操作符可被用於 k-forms 和 k-tensors 例如加法、乘法等，如已定義。
Usage

```r
## S3 method for class 'kform'
Ops(e1, e2 = NULL)
## S3 method for class 'ktensor'
Ops(e1, e2 = NULL)
```

Arguments

- `e1`, `e2` Objects of class `kform` or `ktensor`

Details

The functions `Ops.kform()` and `Ops.ktensor()` pass unary and binary arithmetic operators (”+”, ”-”, ”*”, ”/” and ”^”) to the appropriate specialist function by coercing to spray objects.

For wedge products of \(k\)-forms, use `wedge()` or `%^%` or `^`; and for tensor products of \(k\)-tensors, use `tensorprod()` or `%X%`.

Value

All functions documented here return an object of class `kform` or `ktensor`.

Note

A plain asterisk, “*” behaves differently for ktensors and kforms. Given two ktensors \(T1\), \(T2\), then “\(T1*T2\)" will return the their tensor product. This on the grounds that the idiom has only one natural interpretation. But its use is discouraged (use `%X%` or `tensorprod()` instead). An asterisk can also be used to multiply a tensor by a scalar, as in \(T1*5\).

An asterisk cannot be used to multiply two kforms \(K1\), \(K2\), as in \(K1*K2\), which will always return an error. This on the grounds that it has no sensible interpretation in general and you probably meant to use a wedge product, \(K1^K2\). Note that multiplication by scalars is acceptable, as in \(K1*6\). Further note that \(K1*K2\) returns an error even if one or both is a 0-form (or scalar), as in \(K1*scalar(3)\). This behaviour may change in the future.

In the package the caret (”^”) evaluates the wedge product; note that `%^%` is also acceptable. Powers simply do not make sense for alternating forms: \(S^{^%}S = S^{^%}S\) is zero identically. Here the caret is interpreted consistently as a wedge product, and if one of the factors is numeric it is interpreted as a zero-form (that is, a scalar). Thus \(S^{^%}2 = wedge(S, 2) = 2^{^%}S = S^{^%}2 = S^{^%}S\), and indeed \(S^{^%}n = S^{^%}n\). Caveat emptor! If \(S\) is a kform object, it is very tempting [but incorrect] to interpret “\(S^{^%}3\)” as something like “\(S\) to the power 3”. See also the note at `Ops.clifford` in the `clifford` package.

Powers are not implemented for ktensors on the grounds that a ktensor to the power zero is not defined.

Note that one has to take care with order of operations if we mix `^` with `*`. For example, `dx ^ (6*dy)` is perfectly acceptable; but `(dx ^ 6)*dy` will return an error, as will the unbracketed form `dx ^ 6 * dy`. In the second case we attempt to use an asterisk to multiply two k-forms, which triggers the error.

Author(s)

Robin K. S. Hankin
Examples

```r
## dx_1 ^ dx_2 + 6dx_5 ^ dx_6:
as.kform(1) ^ as.kform(2) + 6*as.kform(5) ^ as.kform(6)

k1 <- kform_general(4,2,rnorm(6))
k2 <- kform_general(4,2,rnorm(6))

E <- matrix(rnorm(8),4,2)
as.function(k1+k2)(E)

## verify linearity, here 2*k1 + 3*k2:
as.function(2*k1+3*k2)(E)-(2*as.function(k1)(E) + 3*as.function(k2)(E))
## should be small
```

print.stokes

Print methods for \textit{k}-tensors and \textit{k}-forms

Description

Print methods for objects with options for printing in matrix form or multivariate polynomial form

Usage

```r
## S3 method for class 'kform'
print(x, ...)
## S3 method for class 'ktensor'
print(x, ...)
```

Arguments

- \textit{x} \hspace{1cm} \textit{k}-form or \textit{k}-tensor
- \textit{...} \hspace{1cm} Further arguments (currently ignored)

Details

The print method is designed to tell the user that an object is a tensor or a \textit{k}-form. It prints a message to this effect (with special dispensation for zero tensors), then calls the spray print method.

Value

Returns its argument invisibly.
Note

The print method asserts that its argument is a map from $V^k$ to $\mathbb{R}$ with $V = \mathbb{R}^n$. Here, $n$ is the largest element in the index matrix. However, such a map naturally furnishes a map from $(\mathbb{R}^m)^k$ to $\mathbb{R}$, provided that $m \geq n$ via the natural projection from $\mathbb{R}^n$ to $\mathbb{R}^m$. Formally this would be $(x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_n, 0, \ldots, 0) \in \mathbb{R}^m$. In the case of the zero $k$-form or $k$-tensor, “$n$” is to be interpreted as “any $n \geq 0$”. See also dovs().

By default, the print method uses the spray print methods, and as such respects the polyform option. However, setting polyform to TRUE can give misleading output, because spray objects are interpreted as multivariate polynomials not differential forms (and in particular uses the caret to signify powers).

It is much better to use options ktensor_symbolic_print or kform_symbolic_print instead. If these options are non-null, the print method uses as.symbolic() to give an alternate way of displaying $k$-tensors and $k$-forms. The generic non-null value would be “x” which gives output like “dx1 ^ dx2”. However, it has two special values: set kform_symbolic_print to “dx” for output like “dx ^ dz” and “txyz” for output like “dt ^ dx”, useful in relativistic physics with a Minkowski metric. See the examples.

More detail is given at symbolic.Rd and the dx vignette.

Author(s)

Robin K. S. Hankin

See Also

as.symbolic, dovs

Examples

```r
a <- rform()
a

options(kform_symbolic_print = "x")
a

options(kform_symbolic_print = "dx")
kform(spray(kform_basis(3,2),1:3))

kform(spray(kform_basis(4,2),1:6)) # runs out of symbols

options(kform_symbolic_print = "txyz")
kform(spray(kform_basis(4,2),1:6)) # standard notation

options(kform_symbolic_print = NULL) # revert to default
a
```
rform

Random kforms and ktensors

Description
Random k-form objects and k-tensors, intended as quick “get you going” examples

Usage
rform(terms=9,k=3,n=7,coeffs,ensure=TRUE)
rtensor(terms=9,k=3,n=7,coeffs)

Arguments
- terms: Number of distinct terms
- k, n: A k-form maps \( V^k \) to \( \mathbb{R} \), where \( V = \mathbb{R}^n \)
- coeffs: The coefficients of the form; if missing use seq_len(terms)
- ensure: Boolean with default TRUE meaning to ensure that the dovs() of the returned value is in fact equal to n. If FALSE, sometimes the dovs() is strictly less than n because of random sampling

Details
What you see is what you get, basically.
Note that argument terms is an upper bound, as the index matrix might contain repeats which are combined.

Value
All functions documented here return an object of class kform or ktensor.

Author(s)
Robin K. S. Hankin

Examples
rform()
rform() ^ rform()
rtensor() %*% rtensor()
rform() ^ dx
rform() ^ dx ^ dy
### Description

Scalars: 0-forms and 0-tensors

### Usage

```r
classical_scalar(s,kform=TRUE,lose=FALSE)
is.classical_scalar(M)
```

### Arguments

- **s**
  - A scalar value; a number

- **kform**
  - Boolean with default TRUE meaning to return a kform and FALSE meaning to return a ktensor

- **M**
  - Object of class ktensor or kform

- **lose**
  - In function classical_scalar(), Boolean with TRUE meaning to return a normal scalar, and default FALSE meaning to return a formal 0-form or 0-tensor

### Details

A k-tensor (including k-forms) maps k vectors to a scalar. If k = 0, then a 0-tensor maps no vectors to a scalar, that is, mapping nothing at all to a scalar, or what normal people would call a plain old scalar. Such forms are created by a couple of constructions in the package, specifically classical_scalar(), kform_general(1,0) and contract(). These functions take a lose argument that behaves much like the drop argument in base extraction. Functions 0form() and 0tensor() are wrappers for classical_scalar().

Function lose() takes an object of class ktensor or kform and, if of arity zero, returns the coefficient.

Note that function kform() always returns a kform object, it never loses attributes.

There is a slight terminological problem. A k-form maps k vectors to the reals: so a 0-form maps 0 vectors to the reals. This is what anyone on the planet would call a scalar. Similarly, a 0-tensor maps 0 vectors to the reals, and so is a scalar. Mathematically, there is no difference between 0-forms and 0-tensors, but the package makes a distinction:

```r
> classical_scalar(5,kform=TRUE)
An alternating linear map from V^0 to R with V=R^0:
```
summary.stokes 29

```r
val = 5
> scalar(5, kform=FALSE)

A linear map from \( V^0 \) to \( R \) with \( V=R^0 \):
val
  = 5
>
```

Compare zero tensors and zero forms. A zero tensor maps \( V^k \) to the real number zero, and a zero form is an alternating tensor mapping \( V^k \) to zero (so a zero tensor is necessarily alternating). See `zero.Rd`.

**Value**

The functions documented here return an object of class `kform` or `ktensor`, except for `is.scalar()`, which returns a Boolean.

**Author(s)**

Robin K. S. Hankin

**See Also**

- `zeroform`

**Examples**

```r
o <- scalar(5)
o
lose(o)

kform_general(1, 0)
kform_general(1, 0, lose=FALSE)
```

---

**summary.stokes**  
**Summaries of tensors and alternating forms**

**Description**

A summary method for tensors and alternating forms, and a print method for summaries.
Usage

```r
## S3 method for class 'kform'
summary(object, ...)
## S3 method for class 'ktensor'
summary(object, ...)
## S3 method for class 'summary.kform'
print(x, ...)
## S3 method for class 'summary.ktensor'
print(x, ...)
```

Arguments

- `object, x` Object of class `ktensor` or `kform`
- `...` Further arguments, passed to `head()`

Details

Summary method for tensors and alternating forms. Uses `spray::summary()`.

Author(s)

Robin K. S. Hankin

Examples

```r
a <- rform(100)
summary(a)
options(kform_symbolic_print = TRUE)
summary(a)
options(kform_symbolic_print = NULL)  # restore default
```

## Symbolic form

<table>
<thead>
<tr>
<th>symbolic</th>
<th>Symbolic form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Returns a character string representing $k$-tensor and $k$-form objects in symbolic form. Used by the print method if either option `kform_symbolic_print` or `ktensor_symbolic_print` is non-null.

Usage

```r
as.symbolic(M, symbols=letters, d="")
```
Arguments

- **M**: Object of class `kform` or `ktensor`; a map from $V^k$ to $\mathbb{R}$, where $V = \mathbb{R}^n$
- **symbols**: A character vector giving the names of the symbols
- **d**: String specifying the appearance of the differential operator

Details

Spivak (p89), in archetypically terse writing, states:

A function $f$ is considered to be a 0-form and $f \cdot \omega$ is also written $f \wedge \omega$. If $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable, then $Df(p) \in \Lambda^1(\mathbb{R}^n)$. By a minor modification we therefore obtain a 1-form $df$, defined by

$$df(p)(v_p) = Df(p)(v).$$

Let us consider in particular the 1-forms $d\pi^i$. It is customary to let $x^i$ denote the function $\pi^i$ (on $\mathbb{R}^3$ we often denote $x^1$, $x^2$, and $x^3$ by $x$, $y$, and $z$). This standard notation has obvious disadvantages but it allows many classical results to be expressed by formulas of equally classical appearance. Since $dx^i(p)(v_p) = d\pi^i(p)(v_p) = D\pi^i(p)(v) = v^i$, we see that $dx^1(p), \ldots, dx^n(p)$ is just the dual basis to $(e_1)_p, \ldots, (e_n)_p$. Thus every k-form $\omega$ can be written

$$\omega = \sum_{i_1 \leq \cdots \leq i_k} \omega_{i_1, \ldots, i_k} dx^{i_1} \wedge \cdots \wedge dx^{i_k}.$$ 

Function `as.symbolic()` uses this format. For completeness, we add (p77) that $k$-tensors may be expressed in the form

$$\sum_{i_1, \ldots, i_k = 1}^{n} a_{i_1, \ldots, i_k} \phi_{i_1} \otimes \cdots \otimes \phi_{i_k}.$$ 

and this form is used for $k$-tensors.

Value

Returns a “noquote” character string.

Author(s)

Robin K. S. Hankin

See Also

- `print.stokes`, `dx`

Examples

```r
(o <- kform_general(3,2,1:3))
as.symbolic(o,d="d",symbols=letters[23:26])

(a <- rform(n=50))
as.symbolic(a,symbols=state.abb)
```
Description

Tensor products of \( k \)-tensors

Usage

\[
tensorprod(U, \ldots) \\
tensorprod2(U1, U2)
\]

Arguments

\( U, U1, U2 \) Object of class \( ktensor \)
\( \ldots \) Further arguments, currently ignored

Details

Given a \( k \)-tensor \( S \) and an \( l \)-tensor \( T \), we can form the tensor product \( S \otimes T \), defined as

\[
S \otimes T (v_1, \ldots, v_k, v_{k+1}, \ldots, v_{k+l}) = S (v_1, \ldots, v_k) \cdot T (v_{k+1}, \ldots, v_{k+l}).
\]

Package idiom for this includes \( tensorprod(S, T) \) and \( S \%X\% T \); note that the tensor product is not commutative. Function \( tensorprod() \) can take any number of arguments (the result is well-defined because the tensor product is associative); it uses \( tensorprod2() \) as a low-level helper function.

Value

The functions documented here all return a spray object.

Note

The binary form \( \%X\% \) uses uppercase \( X \) to avoid clashing with \( \%x\% \) which is the Kronecker product in base \( R \).

Author(s)

Robin K. S. Hankin

References

Spivak 1961

See Also

\( ktensor \)
transform

Examples

(A <- ktensor(spray(matrix(c(1,1,2,2,3,3),2,3,byrow=TRUE),1:2)))
(B <- ktensor(spray(10+matrix(4:9,3,2),5:7)))
tensorprod(A,B)

A %*% B - B %*% A

Va <- matrix(rnorm(9),3,3)
Vb <- matrix(rnorm(38),19,2)

LHS <- as.function(A %*% B)(cbind(rbind(Va,matrix(0,19-3,3)),Vb))
RHS <- as.function(A)(Va) * as.function(B)(Vb)

c(LHS=LHS,RHS=RHS,diff=LHS-RHS)

---

transform

Linear transforms of *k*-forms

Description

Given a *k*-form, express it in terms of linear combinations of the $dx_i$

Usage

- pullback(K,M)
- stretch(K,d)

Arguments

- **K**: Object of class kform
- **M**: Matrix of transformation
- **d**: Numeric vector representing the diagonal elements of a diagonal matrix

Details

Function **pullback()** calculates the pullback of a function. A vignette is provided at ‘pullback.Rmd’.

Suppose we are given a two-form

$$\omega = \sum_{i<j} a_{ij} dx_i \wedge dx_j$$

and relationships
\[ dx_i = \sum_r M_{ir} dy_r \]

then we would have

\[ \omega = \sum_{i<j} a_{ij} \left( \sum_r M_{ir} dy_r \right) \wedge \left( \sum_r M_{jr} dy_r \right). \]

The general situation would be a \( k \)-form where we would have

\[ \omega = \sum_{i_1<\cdots<i_k} a_{i_1\ldots i_k} dx_{i_1} \wedge \cdots \wedge dx_{i_k} \]

giving

\[ \omega = \sum_{i_1<\cdots<i_k} \left[ a_{i_1\ldots i_k} \left( \sum_r M_{i_1 r} dy_r \right) \wedge \cdots \wedge \left( \sum_r M_{i_k r} dy_r \right) \right]. \]

The `transform()` function does all this but it is slow. I am not 100% sure that there isn’t a much more efficient way to do such a transformation. There are a few tests in tests/testthat and a discussion in the stokes vignette.

Function `stretch()` carries out the same operation but for \( M \) a diagonal matrix. It is much faster than `transform()`.

**Value**

The functions documented here return an object of class `kform`.

**Author(s)**

Robin K. S. Hankin

**References**


**See Also**

`wedge`

**Examples**

# Example in the text:
K <- as.kform(matrix(c(1,1,2,3),2,2),c(1,5))
M <- matrix(1:9,3,3)
pullback(K,M)

# Demonstrate that the result can be complicated:
M <- matrix(rnorm(25),5,5)
The Vector cross product

The vector cross product $\mathbf{u} \times \mathbf{v}$ for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ is defined in elementary school as

$$
\mathbf{u} \times \mathbf{v} = (u_2v_3 - u_3v_2, u_2v_3 - u_3v_2, u_2v_3 - u_3v_2).
$$

Function `vcp3()` is a convenience wrapper for this. However, the vector cross product may easily be generalized to a product of $n-1$-tuples of vectors in $\mathbb{R}^n$, given by package function `vector_cross_product()`.

Vignette `vector_cross_product`, supplied with the package, gives an extensive discussion of vector cross products, including formal definitions and verification of identities.

Usage

```r
vector_cross_product(M)
vcp3(u, v)
```

Arguments

- `M`  
  Matrix with one more row than column; columns are interpreted as vectors
- `u, v`  
  Vectors of length 3, representing vectors in $\mathbb{R}^3$
Details

A joint function profile for `vector_cross_product()` and `vcp3()` is given with the package at vignette("vector_cross_product").

Value

Returns a vector

Author(s)

Robin K. S. Hankin

See Also

cross

Examples

```r
vector_cross_product(matrix(1:6,3,2))

M <- matrix(rnorm(30),6,5)
LHS <- hodge(as.1form(M[,1])^as.1form(M[,2])^as.1form(M[,3])^as.1form(M[,4])^as.1form(M[,5]))
RHS <- as.1form(vector_cross_product(M))
LHS-RHS # zero to numerical precision

# Alternatively:
hodge(Reduce(`^`,sapply(seq_len(5),function(i){as.1form(M[,i]),simplify=FALSE))))
```

---

volume

The volume element

Description

The volume element in \( n \) dimensions

Usage

```r
volume(n)
is.volume(K,n=dovs(K))
```

Arguments

- **n**: Dimension of the space
- **K**: Object of class `kform`
Details

Spivak phrases it well (theorem 4.6, page 82):

If $V$ has dimension $n$, it follows that $\Lambda^n(V)$ has dimension 1. Thus all alternating $n$-tensors on $V$ are multiples of any non-zero one. Since the determinant is an example of such a member of $\Lambda^n(V)$ it is not surprising to find it in the following theorem:

Let $v_1, \ldots, v_n$ be a basis for $V$ and let $\omega \in \Lambda^n(V)$. If $w_i = \sum_{j=1}^n a_{ij} v_j$ then

$$\omega(w_1, \ldots, w_n) = \det(a_{ij}) \cdot \omega(v_1, \ldots, v_n)$$

(see the examples for numerical verification of this).

Neither the zero $k$-form, nor scalars, are considered to be a volume element.

Value

Function volume() returns an object of class kform; function is.volume() returns a Boolean.

Author(s)

Robin K. S. Hankin

References

• M. Spivak 1971. Calculus on manifolds, Addison-Wesley

See Also

zeroform,as.1form,dovs

Examples

dx^dy^dz == volume(3)

p <- 1
for(i in 1:7){p <- p ^ as.kform(i)}
p
p == volume(7)  # should be TRUE

o <- volume(5)
M <- matrix(runif(25),5,5)

det(M) - as.function(o)(M)  # should be zero

is.volume(d(1) ^ d(2) ^ d(3) ^ d(4))
is.volume(d(1:9))
Description

Wedge products of \( k \)-forms

Usage

\[
\begin{align*}
\text{wedge2}(K1,K2) \\
\text{wedge}(x, \ldots)
\end{align*}
\]

Arguments

\( K1, K2, x, \ldots \) \( k \)-forms

Details

Wedge product of \( k \)-forms.

Value

The functions documented here return an object of class \( k\text{form} \).

Note

In general use, use \texttt{wedge()} or \texttt{^} or \texttt{\%\%}, as documented under \texttt{Ops}. Function \texttt{wedge()} uses low-level helper function \texttt{wedge2()}, which takes only two arguments.

A short vignette is provided with the package: type \texttt{vignette("wedge")} at the commandline.

Author(s)

Robin K. S. Hankin

See Also

\texttt{Ops}

Examples

\[
\begin{align*}
k1 &\leftarrow \text{as.kform(cbind(1:5,2:6),1:5)} \\
k2 &\leftarrow \text{as.kform(cbind(5:7,6:8,7:9),1:3)} \\
k3 &\leftarrow \text{kform\_general(1:6,2)} \\
\text{a1} &\leftarrow \text{wedge2(k1,wedge2(k2,k3))} \\
\text{a2} &\leftarrow \text{wedge2(wedge2(k1,k2),k3)} \\
\text{is.zero(a1-a2)} &\# \text{ NB terms of a1, a2 in a different order!}
\end{align*}
\]
# This is why wedge(k1, k2, k3) is well-defined. Can also use ^:
# k1 ^ k2 ^ k3

zap

Zap small values in k-forms and k-tensors

Description

Equivalent to zapsmall()

Usage

zap(X)

## S3 method for class 'kform'
zap(X)

## S3 method for class 'ktensor'
zap(X)

Arguments

X Tensor or k-form to be zapped

Details

Given an object of class ktensor or kform, coefficients close to zero are ‘zapped’, i.e., replaced by ‘0’, using base::zapsmall().

Note, zap() actually changes the numeric value, it is not just a print method.

Value

Returns an object of the same class

Author(s)

Robin K. S. Hankin

Examples

S <- rform(7)
S == zap(S)
Zero tensors and zero forms

Description
Correct idiom for generating zero $k$-tensors and $k$-forms

Usage
zeroform(n)
zerotensor(n)
is.zero(x)
is.empty(x)

Arguments
n Arity of the $k$-form or $k$-tensor
x Object to be tested for zero

Value
Returns an object of class kform or ktensor.

Note
Idiom such as as.ktensor(rep(1,n),0) and as.kform(rep(1,5),0) and indeed as.kform(1:5,0) is incorrect as the arity of the tensor is lost.
A 0-form is not the same thing as a zero tensor. A 0-form maps $V^0$ to the reals; a scalar. A zero tensor maps $V^k$ to zero. Some discussion is given at scalar.Rd.

Author(s)
Robin K. S. Hankin

See Also
scalar

Examples
zerotensor(5)
zeroform(3)

x <- rform(k=3)
x*0 == zeroform(3)  # should be true
x  == x + zeroform(3)  # should be true
y <- rtensor(k=3)
y*0 == zerotensor(3)  # should be true
y  == y+zerotensor(3)  # should be true

## Following idiom is plausible but fails because as.ktensor(coeffs=0)
## and as.kform(coeffs=0) do not retain arity:
##
## as.ktensor(1+diag(5)) + as.ktensor(rep(1,5),0)  # fails
## as.kform(matrix(1:6,2,3)) + as.kform(1:3,0)  # also fails
Index

* datasets
  dx, 12
  ex, 13

* package
  stokes-package, 2

* symbolmath
  coeffs, 8
  Ops.kform, 23
  print.stokes, 25
  %x% (tensorprod), 32
  %^% (wedge), 38
  @form(scalar), 28
  @tensor (scalar), 28

Alt, 5, 10
as.1form, 7, 37
as.function.kform (kform), 18
as.function.ktensor (ktensor), 22
as.kform (kform), 18
as.ktensor (ktensor), 22
as.spray (coeffs), 8
as.symbolic, 26
as.symbolic (symbolic), 30

coeff (coeffs), 8
coeffs, 8
coeffs, kform-method (coeffs), 8
coeffs, ktensor-method (coeffs), 8
coeffs.kform (coeffs), 8
coeffs.ktensor (coeffs), 8
coeffs<- (coeffs), 8
coeffs<-, kform-method (coeffs), 8
coeffs<-, ktensor-method (coeffs), 8
coeffs<- .kform (coeffs), 8
coeffs<- .ktensor (coeffs), 8
coeffs<- .spray (coeffs), 8
consolidate, 9
contract, 10
contract_elementary (contract), 10
cross, 36
d, 13, 14
d (kform), 18
discard (keep), 17
dovs, 11, 26, 37
drop (scalar), 28
dx, 12, 31
dy (dx), 12
dz (dx), 12
e (kform), 18
ex, 13
ey (ex), 13
ez (ex), 13
general_kform (kform), 18
grad (as.1form), 7
Hodge (hodge), 14
hodge, 14, 21
include_perms (consolidate), 9
inner, 15
inner_product (inner), 15
is.empty (zero), 40
is.form (kform), 18
is.kform (kform), 18
is.ktensor (ktensor), 22
is.scalar (scalar), 28
is.tensor (ktensor), 22
is.volume (volume), 36
is.zero (zero), 40
issmall, 17
keep, 17
kform, 6, 8, 10, 16, 18, 23
kform_basis (kform), 18
kform_general (kform), 18
kform_symbolic_print (print.stokes), 25
kform_to_ktensor (consolidate), 9
kill_trivial_rows (consolidate), 9
kinner, 21

42