Package ‘wedge’
September 4, 2019

Type Package
Title The Exterior Calculus
Version 1.0-3
Depends spray (>= 1.0-7)
Suggests knitr, Deriv, testthat
VignetteBuilder knitr
Imports permutations (>= 1.0-4), partitions, magrittr, methods
Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>
Description Provides functionality for working with differentials, k-forms, wedge products, Stokes's theorem, and related concepts from the exterior calculus. The canonical reference would be:
License GPL-2
URL https://github.com/RobinHankin/wedge.git
BugReports https://github.com/RobinHankin/wedge/issues
NeedsCompilation no
Author Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)
Repository CRAN
Date/Publication 2019-09-04 05:10:03 UTC

R topics documented:

```
  wedge-package ......................................................... 2
  Alt ................................................................. 4
  as.1form .......................................................... 6
  consolidate ....................................................... 7
  contract ........................................................... 8
  cross ............................................................... 9
  hodge ............................................................... 10
```
Description


Details

The DESCRIPTION file:

Package: wedge
Type: Package
Title: The Exterior Calculus
Version: 1.0-3
Depends: spray (>= 1.0-7)
Suggests: knitr, Deriv, testthat
VignetteBuilder: knitr
Imports: permutations (>= 1.0-4), partitions, magrittr, methods
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 differentials, k-forms, wedge products, Stokes's theorem, and related concepts from the exterior calculus.
License: GPL-2
URL: https://github.com/RobinHankin/wedge.git
BugReports: https://github.com/RobinHankin/wedge/issues
Author: Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Index of help topics:
Alt

Ops.kform

as.1form

consolidate

contract

cross

hodge

inner

issmall

keep

kform

ktensor

rform

scalar

symbolic

transform

volume

wedge

wedge-package

zeroform

Alternating multilinear forms

Arithmetic Ops Group Methods for 'kform' and 'ktensor' objects

Coerce vectors to 1-forms

Various low-level helper functions

Contractions of k-forms

Cross products of k-tensors

Hodge star operator

Inner product operator

Is a form zero to within numerical precision?

Keep or drop variables

k-forms

k-tensors

Random kforms and ktensors

Lose attributes

Symbolic form

Linear transforms of k-forms

The volume element

Wedge products

The Exterior Calculus

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)

NA

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 R \) (here \( V=\mathbb{R}^{15} \)):
as.function(U1)(matrix(rnorm(45),15,3))
```
## Tensor cross-product is cross() 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 %^% (K2 %^% K3)

## k-forms can be coerced to a function and wedge product:
f <- as.function(K1 %^% K2 %^% K3)

## E is a a random point in V^k:
E <- 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 <- as.kform(1)
dy <- as.kform(2)
dz <- as.kform(3)

dx %^% dy %^% dz
K3 %^% dx %^% dy %^% dz

---

**Alt**

**Alternating multilinear forms**

**Description**

Converts a k-tensor to alternating form

**Usage**

Alt(S)
**Arguments**

S  
A multilinear form, an object of class ktensor

**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 \(\text{Alt()}\) takes and returns an object of class ktensor.

**Value**

Returns an alternating \(k\)-tensor. To coerce to a \(k\)-form, which is a much more efficient representation, use \(\text{as.kform()}\).

**Author(s)**

Robin K. S. Hankin

**See Also**

kform

**Examples**

\[
S \leftarrow \text{as.ktensor(expand.grid(1:3,1:3),rnorm(9))}
\]
\[
S
\]
\[
\text{Alt}(S)
\]
\[
\text{issmall}(\text{Alt}(S) - \text{Alt}(\text{Alt}(S))) \quad \# \text{ should be TRUE}
\]
Description

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

Usage

\texttt{as.1form(v)}

\texttt{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

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

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

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

then

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

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

Value

A one-form
**consolidate**

**Author(s)**
Robin K. S. Hankin

**See Also**
kform

**Examples**

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

as.1form(rnorm(20))

grad(c(4,7)) %*% grad(1:4)
```

---

| consolidate | Various low-level helper functions |
---|---|

**Description**

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

**Usage**

```r
consolidate(S)
kill_trivial_rows(S)
include_perms(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
Author(s)
Robin K. S. Hankin

See Also
ktensor, kform

Examples

```r
S <- spray(matrix(c(1,1,2,2,1,3,3,1,3,5),ncol=2,byrow=TRUE),1:5)
kil_trivial_rows(S)
consolidate(S)

## Not run: include_perms(S) # This will fail because of the repeated rows
include_perms(kil_trivial_rows(S)) # This should work
```

## Not run

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

### Description

Given a $k$-form $\phi$ and a vector $v$, the contraction $\phi_v$ of $\phi$ and $v$ is a $k-1$-form with

$$ \phi_v(v^1, \ldots, v^{k-1}) = \phi(v, v^1, \ldots, v^{k-1}) $$

if $k > 1$; we specify $\phi_v = \phi(v)$ if $k = 1$.

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 $v$.

### Usage

```r
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
Cross products of \( k \)-tensors

**Description**

Cross products of \( k \)-tensors

**Usage**

```r
cross(U, ...) cross2(U1,U2)
```

**Arguments**

- \( U,U1,U2 \) Object of class `ktensor`
- ... Further arguments, currently ignored
Details

Given a $k$-tensor object $S$ and an $l$-tensor $T$, we can form the cross 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 `cross(S, T)` and `S %X% T`; note that the cross product is not commutative. Function `cross()` can take any number of arguments (the result is well-defined because the cross product is associative); it uses `cross2()` as a low-level helper function.

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`

Examples

```r
M <- cbind(1:4, 2:5)
U1 <- as.ktensor(M, rnorm(4))
U2 <- as.ktensor(t(M), 1:2)

cross(U1, U2)
cross(U2, U1)  # not the same!
U1 %X% U2 - U2 %X% U1
```

---

hodge  

Hodge star operator

Description

Given a $k$-form, return its Hodge dual

Usage

```r
hodge(K, n=max(index(K)), g=rep(1, n), lose=TRUE)
```
inner

Arguments

- **K**: Object of class `kform`
- **n**: Dimensionality of space, defaulting the the largest element of the index
- **g**: Diagonal of the metric tensor, defaulting to the standard metric
- **lose**: Boolean, with default `TRUE` meaning to coerce to a scalar if appropriate

Value

Returns a \((n - k)\)-form

Author(s)

Robin K. S. Hankin

See Also

- `wedge`

Examples

```r
hodge(rform())

hodge(kform_general(4, 2), g = c(-1, 1, 1, 1))

## Some edge-cases:
hodge(zero(5), 9)
hodge(volume(5))
hodge(volume(5), lose = TRUE)
hodge(scalar(7), n = 9)
```

Description

The inner product

Usage

```r
inner(M)
```

Arguments

- **M**: square matrix
Details

The inner product of two vectors \( \mathbf{x} \) and \( \mathbf{y} \) is usually written \( \langle \mathbf{x}, \mathbf{y} \rangle \) or \( \mathbf{x} \cdot \mathbf{y} \), but the most general form would be \( \mathbf{x}^T M \mathbf{y} \) where \( M \) is a positive-definite matrix. Noting that inner products are symmetric, that is \( \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle \) (we are considering the real case only), and multilinear, that is \( \langle \mathbf{x}, a\mathbf{y} + b\mathbf{z} \rangle = a\langle \mathbf{x}, \mathbf{y} \rangle + b\langle \mathbf{x}, \mathbf{z} \rangle \), we see that the inner product is indeed a multilinear map, that is, a tensor.

Function \( \text{inner}(m) \) returns the 2-form that maps \( \mathbf{x}, \mathbf{y} \) to \( \mathbf{x}^T M \mathbf{y} \).

Value

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

Author(s)

Robin K. S. Hankin

See Also

\( k\text{form} \)

Examples

\begin{verbatim}
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)
\end{verbatim}

issmall

Is a form zero to within numerical precision?

Description

Given a \( k \)-form, return \texttt{TRUE} if it is “small”

Usage

\texttt{issmall(M, tol=1e-8)}
**keep**

**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**

```r
o <- kform_general(4,2,runif(6))
M <- matrix(rnorm(36),6,6)

discrepancy <- o - transform(transform(o,M),solve(M))

issmall(discrepancy) # should be TRUE
is.zero(discrepancy) # might be FALSE
```

---

**keep**  

*Keep or drop variables*

**Description**

Keep or drop variables

**Usage**

```r
keep(K, yes)
discard(K, no)
```

**Arguments**

- **K**  
  Object of class `kform`
- **yes, no**  
  Specification of dimensions to either keep (yes) or discard (no), coerced to a free object

**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.
**kform**

**Author(s)**
Robin K. S. Hankin

**See Also**
lose

**Examples**

```r
g 7 3
keep(kform_general(7,3),1:4)  # keeps only terms with dimensions 1-4
discard(kform_general(7,3),1)  # loses any term with a "1" in the index
```

---

**kform**  

**k-forms**

**Description**

Functionality for dealing with $k$-forms

**Usage**

```r
kform(S)
as.kform(M,coeffs,lose=TRUE)
kform_basis(n,k)
kform_general(W,k,coeffs,lose=TRUE)
## S3 method for class 'kform'
as.function(x,...)
```

**Arguments**

- `n`  
  Dimension of the vector space $V = R^n$
- `k`  
  A $k$-form maps $V^k$ to $R$
- `W`  
  Integer vector of dimensions
- `M`  
  Index matrix for a $k$-form
- `coeffs`  
  Coefficients of the $k$-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.

Recall that a *k*-tensor is a multilinear map from $V^k$ to the reals, where $V = R^n$ is a vector space. A multilinear *k*-tensor $T$ is *alternating* if it satisfies

$$T(v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k) = T(v_1, \ldots, v_j, \ldots, v_i, \ldots, v_k)$$

Function `kform_basis()` is a low-level helper function that returns a matrix whose rows constitute a basis for the vector space $\Lambda^k(R^n)$ of *k*-tensors:

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

and in fact

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

where $e_j, 1 \leq j \leq k$ is a basis for $V$.

In the *wedge* package, *k*-forms are represented as sparse arrays (spray objects), but with a class of `c("kform", "spray")`. The constructor function (`kform()`) ensures that rows of the index matrix are strictly nonnegative integers, have no repeated entries, and are strictly increasing.

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

```r
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)
wedge(K1,K2)
```
\begin{quote}
\texttt{f <- as.function(wedge(K1,K2))}
\texttt{E <- matrix(rnorm(32),8,4)}
\texttt{f(E) + f(E[,c(1,3,2,4)])} \# should be zero
\end{quote}

---

\textbf{ktensor} \quad \textit{k-tensors}

\section*{Description}

Functional property for \textit{k}-tensors

\section*{Usage}

\begin{verbatim}
ktensor(S)
as.ktensor(M,coeffs)
## S3 method for class 'ktensor'
as.function(x,...)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
\item \texttt{M,coeffs} \quad Matrix of indices and coefficients, as in \texttt{spray(M,coeffs)}
\item \texttt{S} \quad Object of class \texttt{spray}
\item \texttt{x} \quad Object of class \texttt{ktensor}
\item \texttt{...} \quad Further arguments, currently ignored
\end{itemize}

\section*{Details}

A \textit{k}-tensor object \textit{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

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

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

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

**Author(s)**
Robin K. S. Hankin

**References**
Spivak 1961

**See Also**
cross, kform, wedge

**Examples**

ktensor(rspray(4, powers=1:4))
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!
```
Arithmetic Ops Group Methods for kform and ktensor objects

Description

Allows arithmetic operators to be used for k-forms and k-tensors such as addition, multiplication, etc, where defined.

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 `%%`; and for cross products of k-tensors, use `cross()` or `%X%`.

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)

as.function(2*k1+3*k2)(E) -(2*as.function(k1)(E) + 3*as.function(k1)(E))
## should be small
```
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)
rtensor(terms=9,k=3,n=7,coeffs)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>terms</td>
<td>Number of distinct terms</td>
</tr>
<tr>
<td>k, n</td>
<td>A k-form maps V^k to R, where V = R^n</td>
</tr>
<tr>
<td>coeffs</td>
<td>The coefficients of the form; if missing use 1 (inherited from spray())</td>
</tr>
</tbody>
</table>

Details

What you see is what you get, basically.

Note that argument terms is an upper bound, as the index matrix might contain repeats. But coeffs should have length equal to terms (or 1).

Author(s)

Robin K. S. Hankin

Examples

rform()
rform(coeffs=1:9)  # any repeated rows are combined
dx <- as.kform(1)
dy <- as.kform(2)
rform() %*% dx
rform() %*% dx %*% dy

rtensor()
scalar

Lose attributes

Description

Scalars: 0-forms and 0-tensors

Usage

```r
scalar(s, lose=FALSE)
is.scalar(M)
'0form'(s, lose=FALSE)
## S3 method for class 'kform'
lose(M)
## S3 method for class 'ktensor'
lose(M)
```

Arguments

- **s**: A scalar value; a number
- **M**: Object of class `ktensor` or `kform`
- **lose**: In function `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 `scalar()`, `kform_general(1,0)` and `contract()`. These functions take a `lose` argument that behaves much like the `drop` argument in base extraction.

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.

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.

Author(s)

Robin K. S. Hankin

See Also

`zeroform`, `lose`
Examples

```r
o <- scalar(5)
lose(o)
kform_general(1,0)
kform_general(1,0,lose=FALSE)
```

<table>
<thead>
<tr>
<th>symbolic</th>
<th>Symbolic form</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>lose(o)</td>
</tr>
<tr>
<td>kform_general(1,0)</td>
<td>kform_general(1,0,lose=FALSE)</td>
</tr>
</tbody>
</table>

Description

Prints $k$-tensor and $k$-form objects in symbolic form

Usage

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

Arguments

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

Author(s)

Robin K. S. Hankin

Examples

```r
as.symbolic(rtensor())
as.symbolic(rform())
as.symbolic(kform_general(3,2,1:3),d="d",symbols=letters[23:26])
```
transform

Linear transforms of k-forms

Description

Given a $k$-form, express it in terms of linear combinations of the $dx^i$

Usage

transform(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

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

$$\omega = \sum_{i_1 < \cdots < i_k} a_{i_1 \cdots i_k} dx_{i_1} \wedge \cdots \wedge dx_{i_k}$$

giving

$$\omega = \sum_{i_1 < \cdots < i_k} \left[ a_{i_1 \cdots i_k} \left( \sum_r M_{i_1 r} dy_r \right) \wedge \cdots \wedge \left( \sum_r M_{i_k r} dy_r \right) \right]$$
So $\omega$ was given in terms of $dx_1, \ldots, dx_k$ and we have expressed it in terms of $dy_1, \ldots, dy_k$. So for example if

$$\omega = dx_1 \wedge dx_2 + 5dx_1 \wedge dx_3$$

and

$$\begin{pmatrix} dx_1 \\ dx_2 \\ dx_3 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \begin{pmatrix} dy_1 \\ dy_2 \\ dy_3 \end{pmatrix}$$

then

$$\omega = (dy_1 + 4dy_2 + 7dy_3) \wedge (2dy_1 + 5dy_2 + 8dy_3) + 5(dy_1 + 4dy_2 + 7dy_3) \wedge (3dy_1 + 6dy_2 + 9dy_3)$$

$$= 2dy_1 \wedge dy_1 + 5dy_1 \wedge dy_2 + \cdots + 5 \cdot 7 \cdot 6dx_3 \wedge dx_2 + 5 \cdot 7 \cdot 9dx_3 \wedge dx_3 +$$

$$= -33dy_1 \wedge dy_2 - 66dy_1 \wedge dy_3 - 33dy_2 \wedge dy_3$$

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`.

Function `stretch()` carries out the same operation but for a matrix with zero off-diagonal elements. It is much faster.

**Value**

Returns a $k$-form

**Author(s)**

Robin K. S. Hankin

**References**


**See Also**

`wedge`

**Examples**

```r
t # Example in the text:
k <- as.kform(matrix(c(1,1,2,3),2,2),c(1,5))
m <- matrix(1:9,3,3)
transform(k,m)

t # Demonstrate that the result can be complicated:
m <- matrix(rnorm(25),5,5)
transform(as.kform(1:2),m)
```
# Numerical verification:
o <- rform(terms=2,n=5)
o2 <- transform(transform(o,M),solve(M))
max(abs(value(o-o2))) # zero to numerical precision

# Following should be zero:
transform(as.kform(1),M)-as.kform(matrix(1:5),c(crossprod(M,c(1,rep(0,4))))))

# Following should be TRUE:
issmall(transform(o,crossprod(matrix(rnorm(10),2,5))))

# Some stretch() use-cases:
p <- rform()
p
stretch(p,seq_len(5))
stretch(p,c(1,0,1,1,1)) # kills dimension 2

# Works nicely with pipes:
## Not run:
max(abs(value(o-o %>% transform(M) %>% transform(solve(M)))))
## End(Not run)

---

**volume**

*The volume element*

### Description

The volume element in \( n \) dimensions

### Usage

```r
volume(n)
is.volume(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.

Author(s)
Robin K. S. Hankin

References
Spivak

See Also
zeroform, as.1form

Examples

\[
\text{as.kform(1) \%^% as.kform(2) \%^% as.kform(3) == volume(3) \ # should be TRUE}
\]

\[
o <- \text{volume(5)}
M <- \text{matrix(runif(25),5,5)}
det(M) - \text{as.function(o)(M) \ # should be zero}
\]
Value

Returns a $k$-form.

Note

In general use, use wedge() or %^%. Function wedge() uses low-level helper function wedge2(), which takes only two arguments.

Author(s)

Robin K. S. Hankin

Examples

```r
k1 <- as.kform(cbind(1:5,2:6),1:5)
k2 <- as.kform(cbind(5:7,6:8,7:9),1:3)
k3 <- kform_general(1:6,2)

a1 <- wedge2(k1,wedge2(k2,k3))
a2 <- wedge2(wedge2(k1,k2),k3)

is.zero(a1-a2) # NB terms of a1, a2 in a different order!

# This is why wedge(k1,k2,k3) is well-defined. Can also use %^%:
k1 %^% k2 %^% k3
```

Description

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

Usage

```r
zeroform(n)
zerotensor(n)
```

Arguments

- `n` Arity of the $k$-form or $k$-tensor

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.
Author(s)
Robin K. S. Hankin

See Also
scalar

Examples

\[
\text{as.ktensor(1+diag(5)) + zerotensor(5)} \\
\text{as.kform(matrix(1:6,2,3)) + zeroform(3)}
\]

## Not run:
\[
\text{as.ktensor(1+diag(5)) + as.ktensor(rep(1,5),0) # fails} \\
\text{as.kform(matrix(1:6,2,3)) + as.kform(1:3,0) # also fails}
\]

## End(Not run)
Index

*Topic package
  wedge-package, 2

*Topic symbolmath
  Ops.kform, 18
  %%(cross), 9
  %%(wedge), 25
  0form(scalar), 20

Alt, 4
as.1form, 6, 25
as.function.kform(kform), 14
as.kform(kform), 14
as.ktensor(ktensor), 16
as.symbolic(symbolic), 21

consolidate, 7
contract, 8
contract_elementary(contract), 8
cross, 9, 17
cross2(cross), 9
discard(keep), 13
drop(scalar), 20
drop.free(keep), 13
general_kform(kform), 14
grad(as.1form), 6

Hodge(hodge), 10
hodge, 10

include_perms(consolidate), 7
inner, 11
inner_product(inner), 11
is.scalar(scalar), 20
is.volume(volume), 24
issmall, 12

keep, 13
kform, 5, 7, 8, 12, 14, 17

kform_basis(kform), 14
kform_general(kform), 14
kill_trivial_rows(consolidate), 7
ktensor, 8, 10, 15, 16

lose, 9, 14, 15, 20
lose(scalar), 20
lose_repeats(consolidate), 7

Ops(Ops.kform), 18
Ops.kform, 18

pull-back(transform), 22
pullback(transform), 22
push-forward(transform), 22
pushforward(transform), 22

retain(keep), 13
rform, 19
rkform(rform), 19
rktensor(rform), 19
rtensor(rform), 19

scalar, 20, 27
spray, 3
star(hodge), 10
stretch(transform), 22
symbolic, 21

transform, 22

volume, 24

wedge, 9, 11, 17, 23, 25
wedge-package, 2
wedge2(wedge), 25

zero, 26
zeroform, 20, 25
zeroform(zero), 26
zerotensor(zero), 26