Package ‘multipol’

February 20, 2015

Type Package
Title multivariate polynomials
Version 1.0-6
Date 2008-04-21
Author Robin K. S. Hankin
Depends abind
Suggests polynom
Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>
Description Various utilities to manipulate multivariate polynomials
License GPL
Repository CRAN
Date/Publication 2013-01-21 12:29:37
NeedsCompilation no

R topics documented:

multipol-package .......................................................... 2
as.array ................................................................. 3
as.function.multipol .................................................... 3
constant ................................................................. 4
deriv ................................................................. 6
Extract.multipol .......................................................... 7
is.constant ............................................................... 8
multipol ................................................................. 8
oom ................................................................. 9
Ops.multipol ............................................................ 11
polyprod ............................................................... 12
print.multipol .......................................................... 13
put ............................................................... 14
trim ............................................................... 15

Index 17
Description

Various tools to manipulate and combine multivariate polynomials

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>multipol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Date</td>
<td>2008-01-24</td>
</tr>
<tr>
<td>License</td>
<td>GPL</td>
</tr>
</tbody>
</table>

Basically, coerce an array to a multivariate polynomial (a “multipol”) using `as.multipol()`.

Taking a matrix `a` as an example, because this has two dimensions it may be viewed as a bivariate polynomial with `a[i,j]` being the coefficient of `x^i y^j`. Note the off-by-one issue; see `?Extract`.

Multivariate polynomials of arbitrary arity are a straightforward generalization using appropriately dimensioned arrays.

Arithmetic operations `+`, `-`, `*`, `^` operate as though their arguments are multivariate polynomials.

Even quite small multipols are computationally intense; many coefficients have to be calculated and each is the sum of many terms.

The package would benefit **enormously** by being able to use a sparse array class.

Author(s)

Robin K. S. Hankin

Maintainer: `<r.hankin@noc.soton.ac.uk>`

References

none really

Examples

```
ones(2)*linear(c(1,-1))  # x^2-y^2
ones(2)*(ones(2,2)-uni(2))  # x^3+y^3

a <- as.multipol(matrix(1:12,3,4))
a
```
\begin{verbatim}
a[1,1] <- 11
f <- as.function(a*a)
f(c(1, pi))
\end{verbatim}

---

**as.array**  
*Coerce multipols to arrays*

**Description**  
Coerce multipols to arrays; unclass

**Usage**  
\begin{verbatim}
## S3 method for class 'multipol'
as.array(x, ...)
\end{verbatim}

**Arguments**  
\begin{itemize}
  \item \textbf{x} multipol
  \item \textbf{...} Further arguments passed to \texttt{NextMethod()}
\end{itemize}

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

**Examples**  
a <- as.multipol(matrix(1, 2, 2))
as.array(a)

---

**as.function.multipol**  
*Coerce a multipol to a function*

**Description**  
Coerce a multipol to a function using environments

**Usage**  
\begin{verbatim}
## S3 method for class 'multipol'
as.function(x, ...)
\end{verbatim}
Arguments

x A multipol
... Further arguments, currently ignored

Author(s)

Robin K. S. Hankin

See Also

as.multipol

Examples

a <- as.multipol(array(1:12, c(2,3,2)))

f1 <- as.function(a)
f2 <- as.function(a*a)

x <- matrix(rnorm(15), ncol=3)

f1(x)^2 - f2(x) #should be zero [non-trivial!]

constant Various useful multivariate polynomials

Description

Various useful multivariate polynomials such as homogeneous polynomials, linear polynomials, etc

Usage

constant(d)
product(x)
homog(d, n = d, value = 1)
linear(x, power = 1)
lone(d, x)
single(d, e, power = 1)
uni(d)
zero(d)

Arguments

d Integer giving the dimensionality (arity) of the result
x A vector of integers
n,e,power Integers
value Value for linear multivariate polynomial
Details

In the following, all multipols have their nonzero entries 1 unless otherwise stated.

- Function constant(d) returns the constant multivariate polynomial of arity d.
- Function product(x) returns a multipol of arity length(x) where all(dim(product(x))==x) with all zero entries except the one corresponding to \( \prod_{i=1}^{d} x_i^2[i] \).
- Function homog(d,n) returns the homogeneous multipol of arity d and power n. The coefficients are set to value (default 1); standard recycling is used.
- Function linear(x) returns a multipol of arity length(x) which is linear in all its arguments and whose coefficients are the elements of x. Argument power returns an equivalent multipol linear in x^power.
- Function lone(d,x) returns a multipol of arity d that is a product of variables x[i].
- Function single(d,e,power) returns a multipol of arity d with a single nonzero entry corresponding to dimension e raised to the power power.
- Function uni(d) returns \( x_1 \times x_2 \times \ldots \times x_d \) [it is a convenience wrapper for product(rep(1,d))].
- Function zero(d) returns the zero multipol of arity d [it is a convenience wrapper for 0*constant(d)].
- Function ones(d) returns \( x_1 + x_2 + \ldots + x_d \) [it is a convenience wrapper for linear(rep(1,d))].

Note

In many ways, the functions documented in this section are an advertisement for the inefficiency of dealing with multipols using arrays: sparse arrays would be the natural solution.

Author(s)

Robin K. S. Hankin

See Also

outer, product, is.constant

Examples

```r
product(c(1,2,5))  # x * y^2 * z^5
uni(3)             # xyz
single(3,1)        # x
single(3,2)        # y
single(3,3)        # z
single(3,1,6)      # x^6
single(3,2,6)      # y^6
lone(3,1:2)        # xy
lone(3,c(1,3))     # xz
linear(c(1,2,5))   # x + 2y + 5z
ones(3)            # x+y+z
constant(3)        # 1 + 0x + 0y + 0z
zero(3)            # 0 + 0x + 0y + 0z
homog(3,2)         # x^2 + y^2 + z^2 + xy + xz + yz
```
# now some multivariate factorization:

```r
ones(2)*linear(c(1,-1))  # x^2-y^2
ones(2)*(linear(c(1,1),2)-uni(2))  # x^3+y^3
linear(c(1,-1))*homog(2,2)  # x^3+y^3 again
ones(2)*(ones(2,4)+uni(2)*2-product(c(1,3))-product(c(3,1)))  # x^5+y^5
ones(2)*homog(2,4,c(1,-1,-1,-1,1))  # x^5+y^5 again
```

---

**deriv**

*Partial differentiation*

**Description**

Partial differentiation with respect to any variable

**Usage**

```r
## S3 method for class 'multipol'
deriv(expr, i, derivative = 1, ...)
```

**Arguments**

- `expr`: A multipol
- `i`: Dimension to differentiate with respect to
- `derivative`: How many times to differentiate
- `...`: Further arguments, currently ignored

**Author(s)**

Robin K. S. Hankin

**See Also**

`substitute`

**Examples**

```r
a <- as.multipol(matrix(1:12,3,4))
deriv(a,1)  # standard usage: differentiate WRT x1
deriv(a,2)  # differentiate WRT x2
deriv(a,1,2)  # second derivative
deriv(a,1,3)  # third derivative (zero multipol)
```
Extract or Replace Parts of a multipol

Description

Extract or replace subsets of multipols

Usage

```r
## S3 method for class 'multipol'
x[...]
## S3 replacement method for class 'multipol'
x[...] <- value
```

Arguments

- `x`: A multipol
- `...`: Indices to replace. **Offset zero! See details section**
- `value`: replacement value

Details

Extraction and replacement operate with offset zero (using functions taken from the `Oarray` package); see the examples section. This is so that the index matches the power required (there is an off-by-one issue. The first element corresponds to the zeroth power. One wants index 1 to extract/replace the i-th power and in particular one wants index 0 to extract/replace the zeroth power).

Replacement operators return a multipol. Extraction returns an array. This is because it is often not clear exactly what multipol is desired from an extraction operation (it is also consistent with `Oarray`’s behaviour).

Author(s)

Original code taken from the Oarray package by Jonty Rougier

References


Examples

```r
a <- as.multipol(matrix(1:4,6))
a[2,2] <- 100
a # coefficient of x1^2.x2^2 is 100
a[1:2,1:2] # a matrix. Note this corresponds to first and second powers
# not zeroth and first (what multipol would you want here?)
```
is.constant  

Is a multivariate polynomial constant or zero?

Description

Is a multivariate polynomial constant or zero?

Usage

is.constant(a, allow.untrimmed = TRUE)

is.zero(a, allow.untrimmed = TRUE)

Arguments

a

A multipol

allow.untrimmed

Boolean with default TRUE meaning to allow a multipol to be zero/constant even if one or more array extents exceed 2

Author(s)

Robin K. S. Hankin

See Also

constant

Examples

is.zero(linear(c(1,1i))*linear(c(1,-1i)) - ones(2,2))  # factorize x^2+y^2

multipol  

Coerce and test for multipols

Description

Coerce and test for multipols

Usage

multipol(x)

as.multipol(x)

is.multipol(x)
Arguments
x Object to be coerced to multipol

Details
The usual case is to coerce an array to a multipol. A character string may be given to as.multipol(), which will attempt to coerce to a multipol.

Note
Subsets of a multipol are accessed and set using Oarray-style extraction with an offset of zero.

Author(s)
Robin K. S. Hankin

See Also
extract.multipol

Examples
a <- as.multipol(array(1:12,c(2,3,2)))

Description
Uses Taylor’s theorem to give one over one minus a multipol

Usage
ooom(n, a, maxorder=NULL)

Arguments
n The order of the approximation; see details
a A multipol
maxorder A vector of integers giving the maximum order as per taylor()
Details

The motivation for this function is the formal power series \((1 - x)^{-1} = 1 + x + x^2 + \ldots\). The way to think about it is to observe that \((1 + x + x^2 + \ldots + x^n)(1 - x) = 1 - x^{n+1},\) even if \(x\) is a multivariate polynomial (one needs only power associativity and a distributivity law, so this works for polynomials). The right hand side is 1 if we neglect powers of \(x\) greater than the \(n\)-th, so the two terms on the left hand side are multiplicative inverses of one another.

Argument \(n\) specifies how many terms of the series to take.

The function uses an efficient array method when \(x\) has only a single non-zero entry. In other cases, a variant of Horner’s method is used.

Author(s)

Robin K. S. Hankin

References


See Also

taylor

Examples

\texttt{ooom(4,\text{homog}(3,1))}

# How many 2x2 contingency tables of nonnegative integers with rowsums =
# c(2,2) and colsums = c(2,2) are there? Good gives:

(  
  ooom(2,\text{lone}(4,c(1,3))) *
  ooom(2,\text{lone}(4,c(1,4))) *
  ooom(2,\text{lone}(4,c(2,3))) *
  ooom(2,\text{lone}(4,c(2,4)))
)

# easier to use the aylmer package:

## not run:
library(aylmer)
no.of.boards(matrix(1,2,2))

## End(Not run)
Description

Allows arithmetic operators to be used for multivariate polynomials such as addition, multiplication, and integer powers.

Usage

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

- `mprod(..., trim = TRUE, maxorder=NULL)`
- `mplus(..., trim = TRUE, maxorder=NULL)`
- `mneg(a, trim = TRUE, maxorder=NULL)`
- `mps(a, b, trim = TRUE, maxorder=NULL)`
- `mpow(a, n, trim = TRUE, maxorder=NULL)`

Arguments

- `e1, e2, a`  Multipols; scalars coerced
- `b`  Scalar
- `n`  Integer power
- `...`  Multipols
- `trim`  Boolean, with default TRUE meaning to return a trim()-ed multipol and FALSE meaning not to trim
- `maxorder`  Numeric vector indicating maximum orders of the output [that is, the highest power retained in the multivariate Taylor expansion about rep(0,d)]. Length-one input is recycled to length d; default value of NULL means to return the full result. More details given under taylor()

Details

The function `Ops.multipol()` passes unary and binary arithmetic operators ("+", "+", "+", and "**") to the appropriate specialist function.

In `multipol.R`, these specialist functions all have formal names such as `.multipol.prod.scalar()` which follow a rigorous pattern; they are not intended for the end user. They are not exported from the namespace as they begin with a dot.

Five conveniently-named functions are provided in the package for the end-user; these offer greater control than the arithmetic command-line operations in that arguments `trim` or `maxorder` may be set. They are:

- `mprod()` for products,
- `mplus()` for addition,
- `mneg()` for the negative,
• mps() for adding a scalar,
• mpow() for powers.

Addition and multiplication of multivariate polynomials is commutative and associative, to machine precision.

Author(s)
Robin K. S. Hankin

See Also
outer, trim, taylor

Examples

```r
a <- as.multipol(matrix(1,4,5))
t00+a

f <- as.function(a+1i)
f(5:6)

b <- as.multipol(array(rnorm(12),c(2,3,2)))

f1 <- as.function(b)
f2 <- as.function(b*b)
f3 <- as.function(b^3)  # could have said b*b*b

x <- c(1,pi,exp(1))

f1(x)^2 - f2(x)  #should be zero
f1(x)^3 - f3(x)  #should be zero

x1 <- as.multipol(matrix(1:10,ncol=2))
x2 <- as.multipol(matrix(1:10,nrow=2))
x1+x2
```

---

### polyprod

**Multivariate polynomial product**

**Description**

Gives an generalized outer product of two multipols

**Usage**

```r
polyprod(m1, m2, overlap = 0)
```
Arguments

- `m1, m2`: multipols to be combined
- `overlap`: Integer indicating how many variables are common to `m1` and `m2`; default of zero corresponds to no variables in common

Author(s)

Robin K. S. Hankin

See Also

`Ops.multipol`

Examples

```r
a <- as.multipol(matrix(1,2,2))  # 1+x+y+xy
polyprod(a, a)  # (1+x+y+xy)*(1+z+t+zt)  --- offset=0
polyprod(a, a, 1)  # (1+x+y+xy)*(1+y+z+yz)
polyprod(a, a, 2)  # (1+x+y+xy)^2
```

Description

Print methods for multipols

Usage

```r
## S3 method for class 'multipol'
print(x, ...)  
do_dimnames(a, include.square.brackets =getOption("isb"), varname =
getOption("varname"), xyz =getOption("xyz"))

## S3 method for class 'multipol'
as.character(x, ..., xyz =getOption("xyz"), varname =
getOption("varname"))
```

Arguments

- `x`: Multipol or array
- `include.square.brackets`: Boolean with TRUE meaning to, er, include square brackets in the dimnames (eg `[x3]^5`) and default FALSE meaning to omit them (eg `x3^5`)
- `varname`: String to describe root variable name (eg `varname="y"` gives `y3^5` or `[y3]^5`)
xyz

Boolean with default TRUE meaning to represent multipols of dimension \( d \leq 3 \) using \( x, y, \) and \( z \) for the variable names and FALSE meaning to use \( x_1, x_2, x_3 \). This option is ignored if \( d > 3 \); see examples section

Further arguments (currently ignored)

**Details**

The default behaviour of `do_dimnames()` and `as.character()`, and hence the print method for multipols, may be modified by using the `options()` function. See examples section below.

**Author(s)**

Robin K. S. Hankin

**Examples**

```r
ones(2,5)
options("showchars" = TRUE)
ones(2,5)
options("xyz" = FALSE)
ones(2,5)
```

**Description**

Substitute a value for a variable and return a multipol of arity \( d-1 \)

**Usage**

```r
put(a, i, value, keep = TRUE)
```

**Arguments**

- **a**: multipol
- **i**: Dimension to substitute
- **value**: value to substitute for \( x[i] \)
- **keep**: Boolean with default TRUE meaning to retain singleton dimensions and FALSE meaning to drop them

**Author(s)**

Robin K. S. Hankin
trim

See Also

deriv.multipol

Examples

```r
a <- as.multipol(matrix(1:12,3,4))
put(a,1,pi)
put(a,2,pi)

b <- as.multipol(array(1:12,c(3,2,3)))
put(b,2,pi,TRUE)
put(b,2,pi,FALSE)
```

trim  Remove redundant entries from a multipol

Description

Remove redundant entries from a multivariate polynomial: function `trim()` trims the array of non-significant zeroes as far as possible without altering its value as a multipol; function `taylor()` returns the multivariate Taylor expansion to a specified order.

Usage

```r
trim(a)
taylor(a,maxorder=NULL)
```

Arguments

- **a** A multipol
- **maxorder** The multivariate order of the expansion returned; default of `NULL` means to return a unaltered

Value

Returns a multipol

Note

If a is a zero multipol (that is, a multivariate polynomial with all entries zero) of any size, then `trim(a)` is a zero multipol of the same arity as a but with extent 1 in each direction.

Author(s)

Robin K. S. Hankin
See Also

* Ops.multipol

Examples

```r
a <- matrix(0, 7, 7)
a[1:3, 1:4] <- 1:12
a <- as.multipol(a)
a
trim(a)
taylor(a, 2)
```
Index

*Topic array
  as.array, 3
  as.function.multipol, 3
  constant, 4
  deriv, 6
  Extract.multipol, 7
  is.constant, 8
  multipol, 8
  multipol-package, 2
  ooom, 9
  Ops.multipol, 11
  polyprod, 12
  print.multipol, 13
  put, 14
  trim, 15
  [.multipol (Extract.multipol), 7
  [<-.multipol (Extract.multipol), 7

as.array, 3
as.character (print.multipol), 13
as.function.multipol, 3
as.multipol, 4
as.multipol (multipol), 8
as_function_multipol
  (as.function.multipol), 3
as_function_multipol_vector
  (as.function.multipol), 3

constant, 4, 8
deriv, 6
deriv.multipol, 15
do_dimnames (print.multipol), 13

Extract.multipol, 7
extract.multipol, 9
extract.multipol (Extract.multipol), 7

homog (constant), 4
is.constant, 5, 8

is.multipol (multipol), 8
is.zero (is.constant), 8

linear (constant), 4
lone (constant), 4

mneg (Ops.multipol), 11
mplus (Ops.multipol), 11
mpow (Ops.multipol), 11
mprod (Ops.multipol), 11
mps (Ops.multipol), 11
multipol, 8
multipol-package, 2

ones (constant), 4
oom, 9
Ops.multipol, 11, 13, 16
outer, 5, 12

polyprod, 12
print.multipol, 13
product, 5
product (constant), 4
put, 14

single (constant), 4
substitute, 6

taylor, 10, 12
taylor (trim), 15
trim, 12, 15

uni (constant), 4
zero (constant), 4