Package ‘polyMatrix’

February 3, 2020

Version 0.2.5

Title Infrastructure for Manipulation Polynomial Matrices

Description
Implementation of class "polyMatrix" for storing a matrix of polynomials and implements basic matrix operations; including a determinant and characteristic polynomial.
It is based on the package 'polynom' and uses a lot of its methods to implement matrix operations.
This package includes 2 methods of triangularization of polynomial matrices:
Extended Euclidean algorithm which is most classical but numerically unstable;
Sylvester algorithm based on LQ decomposition.
Both methods are described in
D. Henrion & M. Sebek, Reliable numerical methods for polynomial matrix triangularization,
and in
Salah Labhalla, Henri Lombardi & Roger Marlin,
Algorithmes de calcul de la reduction de Hermite d'une matrice a coefficients polynomeaux,

Type Package
Imports polynom
License MIT + file LICENSE
Depends R (>= 3.0), stats, MTS
Suggests testthat
Repository CRAN
URL https://github.com/namezys/polymatrix

BugReports https://github.com/namezys/polymatrix/issues

NeedsCompilation no

Author Tamas Prohle [aut],
Peter Prohle [aut],
Nikolai Ryzhkov [aut, cre],
Ildiko Laszlo [aut] (<https://orcid.org/0000-0003-2324-8183>),
Ulas Onat Alakent [ctb]
Maintainer Nikolai Ryzhkov <namezys@gmail.com>
Date/Publication 2020-02-03 06:10:02 UTC

R topics documented:

polyMatrix-package .................................................. 3
CanForm ................................................................. 4
ch2pn ................................................................. 5
charpn ................................................................. 7
coefs ................................................................. 9
CommonPolynom ...................................................... 10
const ............................................................... 12
constConv ............................................................ 13
cycFill ............................................................. 14
degree .............................................................. 15
diag ................................................................. 16
dim ................................................................. 18
is.polyMatrix ........................................................ 19
lead ................................................................. 20
matrixMinMax ....................................................... 21
MTS2pM ............................................................. 22
Ops.polyMatrix ...................................................... 23
pMadj .............................................................. 25
pMbas .............................................................. 26
pMcol .............................................................. 27
pMdet .............................................................. 28
pMdiag ............................................................ 29
pMkmm ............................................................. 30
pMprod ............................................................ 31
pMrow ............................................................. 32
pMsgn ............................................................. 32
pMstr ............................................................. 33
pMsub ............................................................. 34
pn2ch ........................................................... 35
polyMconvert ..................................................... 36
polyMgen ........................................................ 37
polyMul .......................................................... 40
pprt .............................................................. 41
predict.charpn .................................................... 42
predict.polyMatrix ................................................ 43
predict.polynomial ................................................ 44
print ............................................................ 45
proper .......................................................... 47
pVsk ............................................................ 48
symb ............................................................ 49
t ................................................................. 50
tr ................................................................. 51
**polyMatrix-package**

Implementation of class "polyMatrix" for storing a matrix of polynomials and implements basic matrix operations; including a determinant and characteristic polynomial. It is based on the package 'polynom' and uses a lot of its methods to implement matrix operations. This package includes 2 methods of triangularization of polynomial matrices: Extended Euclidean algorithm which is most classical but numerically unstable; Sylvester algorithm based on LQ decomposition. Both methods are described in D. Henrion & M. Sebek, Reliable numerical methods for polynomial matrix triangularization, IEEE Transactions on Automatic Control (Volume 44, Issue 3, Mar 1999, Pages 497-508) <doi:10.1109/9.751344> and in Salah Labhalla, Henri Lombardi & Roger Marlin, Algorithmes de calcule de la reduction de Hermite d’une matrice a coefficients polynomeaux, Theoretical Computer Science (Volume 161, Issue 1-2, July 1996, Pages 69-92) <doi:10.1016/0304-3975(95)00090-9>.

**Details**

- Package: polyMatrix
- Type: Package
- Version: 0.0.05
- Date: 2016-06-04
- License: GPL-3

Infrastructure for manipulation polynomial matrices.

**Author(s)**

NA

Maintainer: NA

**References**

See other polynomial packages: polynom, PolynomF, MonoPoly, multipol, mpoly, orthopolynom.

**Examples**

```r
# the c("polyMarray", "polyMatrix") class structure
str(polyMgen.a())
```
Conversion of an ARMA representation to a canonical form

Description
The representation of an ARMA processes is unambiguous only in canonical form. The CanForm method, if the necessary parameters are given converts the ARMA representation to echelon, final or scm form.

Usage
CanForm(pM, form)
CanForm.echelon(pM)
CanForm.final(pM)
CanForm.scm(pM)

Arguments
pM an arbitrary pMvarma class object
form an arbitrary or "echelon", "final", "scm" form code

Value
An pMvarma class object in echelon, final or scm form.

Note
This method doesn’t work yet - is not implemented yet.

See Also
polyMatrix-package
Examples

```r
parma <- polyMgen.varma(3,3.rand=TRUE,degree=c(1,1))
parma

CanForm(parma,"final")
CanForm(parma,"echelon")
CanForm(parma,"scm")
```

# clean up
# rm(parma)

---

**ch2pn**

*Converts a character class object to a polynomial class object*

### Description

Receives a character class object – which looks like the print image of a polynomial object – and returns a polynomial class object.

### Usage

```r
ch2pn(chv, symb = "x")
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>chv</code></td>
<td>a character class object like the print image of a polynomial object</td>
</tr>
<tr>
<td><code>symb</code></td>
<td>the required symbol of input polynom string</td>
</tr>
</tbody>
</table>

### Details

This technical subrutin converts a character object to a polynomial object. The "x" is the default symbol in the input string. The output is a polynomial or a list of polynomials, without any defined symbol. The default print.polynomial writes out the output by the symbol "x".

### Value

A polynomial class object or a list of polynomial objects.

### See Also

`pn2ch`, `pprt`
Examples

```r
chx <- "1 + x + x^3 + 2 * x^4 - 4 * x^5 - 15 * x^16"
p <- ch2pn(chx, "x")
# \"polynomial\" (as in package: \"polynom\")
class(p)
# 1 + x + x^3 + 2*x^4 - 4*x^5 - 15*x^16
p
# ditto \lesssim \"x\" is the default symbol
ch2pn(chx)

# the same polynom over \"z\"
chz <- "1 + z + z^3 + 2 * z^4 - 4 * z^5 - 15 * z^16"
# == 1: the only \"x\" power in the input string
ch2pn(chz)
# 1 + x + x^3 + 2*x^4 - 4*x^5 - 15*x^16
ch2pn(chz, "z")

# the same print by polyMatrix::pprt() print utility
pprt(ch2pn(chz, "z"), "x")
# 1 + 5*x - 5*x^2 - 2*x^3 + x^5 + x^6 - x^7
ch2pn("-2 * x^3 - 5 * x^2 + x^5 + x^6 + 5 * x - x^7 + 1")
# 1
ch2pn("-2 * x^3 - 5 * x^2 + x^5 + x^6 + 5 * x - x^7 + 1", "z")

# 2
ch2pn(" 2")
# 0
ch2pn("z^3")
# z^4
ch2pn("z^3", "z")

# 0
ch2pn(" 3 * z ^ 2 + 5 * z^3")
# 4
ch2pn(" 4 + 5 * z^3")
# 6 + 5*x^3
ch2pn(" 6 + 5 * z^3", "z")

# a list of \"polynomial\" class elements: [[1]] x [[2]] 1 + x^2
ch2pn(c(" x ", " 1 + x^2 "))

# col.wise a \"polyMdlist\" \"polyMatrix\" class object
polyMgen.d(2, 2, rawData=ch2pn(c("1", "x^2", "x", "0")))

# symbole given for output
polyMgen.d(2, 2, rawData=ch2pn(c("1", "x^2", "x", "0")), symb="x")

# symbole given for input and output
polyMgen.d(2, 2, rawData=ch2pn(c("1", "x^2", "x", "0"),symb="x"), symb="x")
```
charpn

# clean up
# rm(chz, p, chx)

charpn

The characteristic polynom of a matrix or a polynomial matrix

Description

The characteristic polynom of a polynomial matrix is a polynom with polynomial coefficients.

Usage

charpn(M)

Arguments

M  polynomial matrix object of polyMatrix class

Value

det(sI-M). When the input is a matrix class object then the value is a polynomial object. When
the input is a polyMatrix class object then a value is charpn class object, which is a list of
polynomial objects.

Note

The solution of the algorithm is simple one, a signed sum of products. There are more elegant and
efficient solutions.

See Also

predict.charpn

Examples

M1<-matrix(c(2,1,-1,0),2,2,byrow=TRUE)
class(M1) # "matrix"
pM1<-M2pM(M1,"polyMdlist") # conversion matrix => polyMatrix
class(pM1) # "polyMdlist" "polyMatrix"

M2<-matrix(c(1,0,-1,2,-1,0,1,1),3,3,byrow=TRUE)
class(M2) # "matrix"
pM2<-M2pM(M2,"polyMdlist") # conversion matrix => polyMatrix
class(pM2) # "polyMdlist" "polyMatrix"

pM3<-polyMgen.d(2,2,rawData=ch2pn(c("1","x","x^2","0")))
class(pM3) # "polyMdlist" "polyMatrix"

pM4<-polyMgen.d(2,2,rawData=ch2pn(c("x","1","x^2","0")))
class(pM4) # "polyMdlist" "polyMatrix"

pM5<-polyMgen.d(3,3,rawData=ch2pn(c("x","1","x^2","0","x","1","x^2","0","x^2")))
class(pM5) # "polyMdlist" "polyMatrix"

pM6<-polyMgen.d(3,3,rawData=ch2pn(c("x","1","x^2","0","x^2","0","x^2","0","x^2")))
class(pM6) # "polyMdlist" "polyMatrix"

ls() # M1, pM1, M2, pM2, M3, pM3, pM4, pM5, pM6

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# the eigenvalues of a matrix are
# roots of the characteristic polynomial

pn1 <- charpn(M1) # the characteristic polynom of M1
ev <- eigen(M1)$values
ev # 1 1
predict(pn1,ev) # OK: zero matrix

pn2 <- charpn(M2) # the characteristic polynom of M2
ev <- eigen(M2)$values
round(ev,3) # 1.5+1.323i 1.5-1.323i 0.0+0.000i
round(predict(pn2,ev)) # OK: 0+0i 0+0i 0+0i

# clean up
# rm(pn1, pn2, ev)

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# the characteristic polynomial and
# the Cayley-Hamilton theorem for 'matrix' and 'polyMatrix' class objects

# ---
# M1, pM1: a 2x2 constant matrix

class(M1) # "matrix"
pn1 <- charpn(M1) # the characteristic polynom of M1
predict(pn1,M1) # OK: zero matrix
class(pM1) # "polyMdlist" "polyMatrix"
pnlp <- charpn(pM1) # the characteristic polynom of pM1
predict(pnlp,pM1) # OK: zero matrix

# ---
# M2, pM2: a 3x3 constant matrix

class(M2) # "matrix"
pn2 <- charpn(M2) # the characteristic polynom of M2
predict(pn2,M2) # OK: zero matrix
class(pM2) # "polyMdlist" "polyMatrix"
pn2p<-charpn(pM2) # the characteristic polynom of pM2
predict(pn2p,pM2) # OK: zero matrix
# pM3, pM4: an arbitrary 2x2 polyMatrix

class(pM3) # "polyMlist" "polyMatrix"
pn3p<-charpn(pM3) # the characteristic polynom of pM3
predict(pn3p,pM3) # OK: zero matrix

class(pM4) # "polyMlist" "polyMatrix"
pn4p<-charpn(pM4) # the characteristic polynom of pM4
predict(pn4p,pM4) # OK: zero matrix

# pM5, pM6: an arbitrary 3x3 polyMatrix

class(pM5) # "polyMlist" "polyMatrix"
 pn5p<-charpn(pM5) # the characteristic polynom of pM5
predict(pn5p,pM5) # OK: zero matrix

class(pM6) # "polyMlist" "polyMatrix"
 pn6p<-charpn(pM6) # the characteristic polynom of pM4
predict(pn6p,pM6) # OK: zero matrix

# -----------------------------------------------
ls()
# 1. 2. | 3. 4. | 5. 6. | 7. 8.
# case real const.pn| real const.pn| arb.pn arb.pn| arb.pn arb.pn
# size 2x2 2x2 | 3x3 3x3 | 2x2 2x2 | 3x3 3x3
# matrices: M1 pM1 | M2 pM2 | pM3 pM4 | pM5 pM6
# char.pn: pn1 pn1p | pn2 pn2p | pn3p pn4p | pn5p pn6p
# -----------------------------------------------

# clean up
# rm(M1, M2, pM1, pM2, pM3, pM4, pM5, pM6)
# rm(pn1, pn1p, pn2, pn2p, pn3p, pn4p, pn5p, pn6p)

coeffs

The coefficients of a polynom or polynomial matrix

description

Returns the coefficients of the given polynom or polynomial matrix.

usage

coeffs(p, degree="all")
Arguments

- `p` a polynomial or polyMatrix class object
- `degree` "all" or the serial number of the requested coefficients.

Details

The `coefs` consists of two methods. One for a polynomial objects, the other one for polyMatrix objects.

Value

A vector of the requested coefficients of the input polynom or a list of the requested coefficient matrices of the input polyMatrix.

See Also

`const`, `lead`, `coefs.polynomial`, `coefs.polyMatrix`

Examples

```r
coefs(polynomial::polynomial(12:0))
coefs(polynomial::polynomial(12:0), degree=c(2,3))
coefs(polyMgen.a())
coefs(polyMgen.b())
coefs(polyMgen.c())
coefs(polyMgen.d())
coefs(polyMgen.d(), degree=1)
```

CommonPolynom

**GCD and LCM for polynomial matrices**

Description

The greatest common divisor of polynomial that keeps the elements of the given polynomial matrix.

Usage

```r
## S3 method for class 'polyMatrix'
GCD(x, type=OPERATION_TYPE_TOTAL, ...)
## S3 method for class 'polyMatrix'
LCM(x, type=OPERATION_TYPE_TOTAL, ...)
```

Arguments

- `x` an polyMatrix class object
- `type` calculates the common polynom column-wise (type="col"), row-wise (type="row") or for the total matrix (type="total", by-default)
- `...` additional arguments
Details

Calculates the greatest common divisor or the least common divisor of the total matrix, or row-wise or column-wise, depending on the parameter type.

Possible operator types: OPERATION_TYPE_TOTAL="total", OPERATION_TYPE_COLUMN="col", OPERATION_TYPE_ROW="row"

Value

A polynom matrix which is the greatest common divisor or the least common multiple for all the elements of the given polynomial matrix or for the columns or for the rows, depending on the value of the 'type' parameter.

See Also

pMdet, pMadj

Examples

\[
\begin{align*}
    p1 & \leftarrow \text{ch2pn}("1 - x") \\
    p2 & \leftarrow \text{ch2pn}("x - x^2") \\
    p3 & \leftarrow \text{ch2pn}("1 - x^2") \\
    p4 & \leftarrow \text{ch2pn}("1 + x") \\
    p5 & \leftarrow \text{ch2pn}("1 + 2*x + x^2") \\
    p6 & \leftarrow \text{ch2pn}("1 - 2*x + x^2") \\
    A & \leftarrow \text{polyMgen.d}(2, 3, \text{rawData=list}(p1, p2, p3, p4, p5, p6)) \\
    GCD(A) & \leftarrow 1 \\
    \text{polynom::GCD(l}(p1, p2, p3, p4, p5, p6)) & \leftarrow 1 \\
    \text{GCD(A, "col")} & \leftarrow 1 \\
    \text{list(} \\
        \text{polynom::GCD(l}(p1, p2)), \\
        \text{polynom::GCD(l}(p3, p4)), \\
        \text{polynom::LCM(l}(p5, p6)) \\
    \text{)} & \leftarrow \text{# 1 - x; 1 + x; 1 - 2*x^2 + x^4} \\
    \text{polynom::GCD(l}(p1, p2)) & \leftarrow 1 \\
    \text{polynom::GCD(l}(p3, p4)) & \leftarrow 1 \\
    \text{polynom::LCM(l}(p5, p6)) & \leftarrow 1 \\
    \text{GCD(A, "row")} & \leftarrow 1 \\
    \text{polynom::GCD(l}(p1, p3, p5)) & \leftarrow 1 \\
    \text{polynom::GCD(l}(p2, p4, p6)) & \leftarrow 1 \\
\end{align*}
\]

# ---
const

The constant of a polynom or a polynomial matrix

Description

Returns the constant of a polynom class object or the constant matrix of a polyMatrix class object.

Usage

const(p)

Arguments

p a polynomial or polyMatrix class object

Value

A numeric object, containing the constant of the given polynom, or of a matrix class object which contains the constant matrix of the given polynomial matrix.

See Also

coefs, lead, const.polyMatrix, const.polynomial
Examples

const(polynom::polynomial(0:12))
const(polyMgen.a())
const(polyMgen.b())
const(polyMgen.c())
const(polyMgen.d())

constConv Conversion of constant matrices

Description
Conversion between the 'matrix' and 'polyMatrix' class representation of constant matrices.

Usage
M2pM(m, class=CLASS_MARRAY)
pM2M(pM)

Arguments
m a matrix class object
class the required class of the value: CLASS_MARRAY, CLASS_MBROAD, CLASS_MCELLS, and CLASS_MDLIST
pM a polyMatrix class object

Details
M2pM converts the matrix class objets to a polyMatrix class object. pM2M converts the zero degree polyMatrix class objects to a matrix object.

Value
The M2pM(m) returns the given 'm' matrix in 'polyMatrix' class form. The pM2M(pM) returns the given 'pM' polynomial matrix in 'matrix' class form, if it is a constant matrix, otherwise gives an error message.

See Also
coefs, lead, degree
Examples

```r
m <- matrix(1:12, 3,4)
m
pM <- M2pM(m)
class(pM)
pM

M2pM(m, "polyMarray")
M2pM(m, "polyMbroad")
M2pM(m, "polyMcells")
M2pM(m, "polyMdlist")

# clean up
# rm(m, pM)
```

cycFill  
A vector or a list of a given length, filled cyclically by the given material

Description

Fills an object by the given material

Usage

cycFill(data, size)

Arguments

data the material
size the desired length

Details

The class of the output is the same as the class of the input. The output object is filled cyclically by the given material u.

Value

Depending on the class of the given material, the class of the result is vector or list. The length of the result equals by the value of the size parameter.

See Also

Used in the code of polyMgen.a, polyMgen.d
Examples

```r
u <- c(4, 1, 3, 2)
cycFill(u, 2)
cycFill(u, 4)
cycFill(u, 6)
cycFill(u, 11)
cycFill(u, 12)

u <- as.list(c(4, 1, 3, 2))
cycFill(u, 2)
cycFill(u, 4)
cycFill(u, 6)
cycFill(u, 11)
cycFill(u, 12)

# clean up
# rm(u)
```

---

**degree**

*The degree of a polynomial or a polynomial matrix*

**Description**

The degree of the highest non-zero coefficient of the polynomials.

**Usage**

```r
degree(p, ...)
```

**Arguments**

- `p` a polynomial or polyMatrix class object
- `...` supplementary options (see `degree.polyMatrix`)

**Value**

When the input is a polynomial class object, then `value` is the degree of the polynom.

When the input is a polyMatrix class object, then in case of `method = "matrix"` the value is a matrix by the same size as the input, containing the degrees of the polynomials. In case of `method = "column"` or `"row"` the value are maximums of the columns or rows of the polynom matrix degrees. In case of `method = "default"` it is the maximum of the degree of all polynoms.

**See Also**

`lead, degree.polyMatrix, degree.polynomial`
Examples

```r
p <- polynom::polynomial(0:12)
class(p) # polynom
degree(p)

pM <- polyMgen.d(3,3,rawData=
   ch2pn(c("-3+x^3","2+4*x","x^2","1","2","3+x","2*x","0","2-3*x")))
pM
degree(pM) # the maximum degree
degree(pM,"c") # column wise maximum degree
degree(pM,"r") # row wise maximum degree
degree(pM,"m") # matrix: element wise degree

degree(polyMconvert(pM,"polyMarray"),"m") # polyMarray class
degree(polyMconvert(pM,"polyMbroad"),"m") # polyMbroad class
degree(polyMconvert(pM,"polyMcells"),"m") # polyMcells class
degree(polyMconvert(pM,"polyMdlist"),"m") # polyMdlist class

# clean up
# rm(p, pM)
```

---

```r
diag

Extracts the diagonal of a polyMatrix, or constructs a diagonal polyMatrix
```

Description

If the input is a polyMatrix then the output is its diagonal. If the input is a vector then the output is a diagonal matrix.

Usage

```r
diag(x, nrow=NULL, ncol=NULL, names=NULL, type=CLASS_LIST, ...)
```

Arguments

- **x**: input material for the diagonal matrix
- **nrow**: number of rows in the output object
- **ncol**: number of columns in the output object
- **names**: (when `x` is a matrix) logical indicating if the resulting vector, the diagonal of `x`, should inherit `names` from `dimnames(x)` if available.
- **type**: the required form and class of the result: CLASS_LIST or CLASS_MATRIX
- **...**:
Details
The \textit{diag} has two different applications. When the class of the first input parameter is

- '\texttt{list}', then returns a \texttt{polyMatrix} with the given diagonal and zero off-diagonal entries.
- '\texttt{polyMatrix}', then extracts the diagonal.

Note. In the first case the elements of the list must be \texttt{polynomial} class objects. In the second case the result is a \texttt{list} or \texttt{polyMatrix} class object determined by the value of the \texttt{type} parameter.

Value
The returned value is a

- \texttt{list} class object, when it extracts the diagonal from a \texttt{polyMatrix}, and the value of \texttt{type} isn't '\texttt{polyMatrix}'.
- \texttt{polyMatrix} class object, when builds a matrix from the given diagonal or when returns the diagonal of a \texttt{polyMatrix}, and the \texttt{type=\textquote{\texttt{polyMatrix}}}.

See Also
\texttt{polyMatrix-package}.

Examples

\begin{verbatim}
# ---
# case of real matrices
m<-matrix(1:12,3,4)
m
diag(m) # first type usage
v<-c(1,5,9) # second type usage
diag(v)
diag(v,4)
diag(v,4,2)
diag(v,ncol=4)

# ---
# case of polynomial matrices
# first kind usage: catch the diagonal elements
px<polyMgen.d(3,3,rawData=ch2pn(
    c("-3+z^2","2+4*z","-z^2",
    "1","z","3+z",
    "2*z","0","2-3*z"),"z"),byrow=TRUE)
class(px) # "polyMdlist" "polyMatrix"
px
w<-diag(px) # the default: type="list"
\end{verbatim}
class(w) # "list"
# List of 3
# $ :Class 'polynomial' num [1:3] -3 0 1
# $ :Class 'polynomial' num 2
# $ :Class 'polynomial' num [1:2] 2 -3
w <- diag(px, type = "polyMatrix")
class(w) # "polyMdlist" "polyMatrix"
w

# second kind usage: compose a diagonal polyMatrix
ci <- c(4, 1, 3, 2)
v <- vector("list", length(ci))
for(i in 1:length(ci))
  v[[i]] <- polynom::polynomial(c(rep(0, ci[i]), 1))
class(v) # "list"
v
diag(v)
diag(v, 4)
diag(v, 3)
diag(v, 5)
diag(v, 3, 5)
pd <- diag(v, 4, 5)
class(pd) # "polyMdlist" "polyMatrix"
pd

# clean up
# rm(ci, i, m, pd, px, v, w)

---

dim

The dimension of a polynomial matrix

Description

Returns the value of the $dim element of the given polyMatrix object.

Usage

## S3 method for class 'polyMatrix'
dim(x)

Arguments

x a polyMatrix class object
Details

A method for 
polyMatrix objects. Does not check the validity of the $dim element.

Value

A 2 element vector which contains the dimension parameters of the input polyMatrix.

Examples

dim(matrix(1:12,3,4)) # dim of a real matrix

A <- polyMgen.a()
class(A)# polyMarray
dim(A) # dim of a polyMatrix

B <- polyMconvert(A,"polyMdlist")
class(B)# polyMdlist
dim(B)

dim(polyMgen.a())
dim(polyMgen.b())
dim(polyMgen.c())
dim(polyMgen.d())

# clean up
# rm(A, B)

is.polyMatrix

Is an object of type 'polyMatrix'?

Description

Checks whether the given object is a polyMatrix object or not.

Usage

is.polyMatrix(x)

Arguments

x an R object.

Details

is.polyMatrix returns TRUE if its argument is a polyMatrix (that is, has "polyMatrix" amongst its classes and one of the four sub-classes: "polyMarray", "polyMbroad", "polyMcells", "polyMdlist") and FALSE otherwise.
The leading coefficient of a polynom or polynomial matrix

**Description**

Returns the leading coefficients of the input polynom or polynomial matrix.

**Usage**

```r
lead(p, method=c("matrix","column","row","element"))
```

**Arguments**

- **p**
  - a polynominal or polyMatrix-package class object
- **method**
  - controls the interpretation of the word: "leading", see the details

**Details**

The four possible interpretation of the `lead` for a polynomial matrix are:

- **matrix**: the highest matrix coefficient of the polynomial matrix.
- **column**: taken the polynomial matrix column-wise, the row vector of the highest coefficient of each column.
- **row**: taken the polynomial matrix row-wise, the column vector of the highest coefficient of each row.
- **element**: a matrix formed by the highest coefficients of the polynomial elements of the polynomial matrix.

**Value**

Returns a matrix by the same size as the polynomial matrix.
matrixMinMax

Maximums and minimums for numeric matrices

Description

Calculate the maximums or minimums column-wise or row-wise, depending on the called method.

Usage

\[
\begin{align*}
\text{colMax}\text{(matrix)} & \quad \text{# column maximums} \\
\text{colMin}\text{(matrix)} & \quad \text{# column minimums} \\
\text{rowMax}\text{(matrix)} & \quad \text{# row maximums} \\
\text{rowMin}\text{(matrix)} & \quad \text{# row minimums}
\end{align*}
\]

Arguments

\[
\begin{align*}
\text{matrix} & \quad \text{a matrix class object}
\end{align*}
\]

See Also

coefs, degree, const

Examples

```r
lead(polynom::polynomial(0:12))
pm <- polyMgen.a()
pa <- polyMconvert(pm,"polyMarray")
pb <- polyMconvert(pm,"polyMbroad")
pc <- polyMconvert(pm,"polyMcells")
pd <- polyMconvert(pm,"polyMdlist")
lead(pa)
lead(pb)
lead(pc)
lead(pd)

rawAdat <- ch2pn(c("-3 + x^2","2 + 4*x","-x^2","1","2","3 + x","2*x","0","2 - 3*x" ))
px <- polyMgen.d(3,3,rawData=rawAdat)

# clean up
# rm(rawAdat)
px

lead(px)
lead(px,method="matrix")
lead(px,method="row")
lead(px,method="column")
lead(px,method="element")

# clean up
# rm(pm, pa, pb, pc, pd, px)
```
Value

A vector by the column or row maximums or minimums of the given matrix.

See Also

`colMin`, `rowMax`, `rowMin`

Examples

```r
set.seed(123)
M <- matrix(rpois(12,3),3,4)
M
# 2 5 3 3
# 4 6 5 6
# 2 0 3 3
colMax(M) # c(4,6,5,6)
colMin(M) # c(2,0,3,3)
rowMax(M) # c(5,6,3)
rowMin(M) # c(2,4,0)
# clean up
# rm(M)
```

`MTS2pM`  
*Converts MTS representation of a VARMA process to a polyMatrix representation*

Description

The MTS representation of VARMA process is a list of 13 elements. The necessary elements to represent a VARMA model are the c(2, 3, 4, 5, 8, 11, 12, 13) elements: ARorder, MAorder, cnst, coef, Sigma, Phi, Theta, Ph0. The other five are redundant parameters.

Usage

`MTS2pM(M)`

Arguments

`M`  
an arbitrary VARMA model list-object of the MTS package

Details

MTS (Multivariate Time Series) is a general package for analyzing multivariate linear time series, written by Ruey S. Tsay.

Value

An `pMvarma` class object, equivalent with the `M` input object, created by the package MTS.
See Also

print.pMvarma, print.polyMatrix

Examples

# runs longly
library(MTS)
set.seed(1)
yt <- VARMAsim(300, arlags=1, malags=1,
      phi=matrix(c(0.2, -0.6, 0.3, 1.1), 2, 2),
      theta=matrix(c(-0.5, 0, 0, -0.5), 2, 2),
      sigma=diag(2))$series
M <- VARMA(yt, p=2, q=2, include.mean=TRUE)
M[-c(1, 6, 7, 9, 10)]
pMvarma <- MTS2pM(M)
print(pMvarma, style = "broad", digits = 3)
pMvarma
print(pMvarma, "matrix", digits = 3)

# clean up
# rm(yt, M, pMvarma)

Ops.polyMatrix

Arithmetic Ops Group Methods for polyMatrix class objects

Description

Allows arithmetic operations by left hand side polynomial matrices. Contains operators such as addition, multiplication, division, etc.

Usage

## S3 method for class 'polyMatrix'
Ops(e1, e2)

Arguments

e1  an object of class polyMatrix
e2  an object of class numeric, character or polyMatrix

Value

A c("polyMdlist","polyMatrix") class polynomial matrix which is the result obtained by performing the operation on the two arguments.
Author(s)
Prohle Tamas

See Also
polyMatrix-package, Ops

Examples
X <- polyMgen.d(2,2, rawData=ch2pn(c("1","1+x^2","x","0")))
Y <- polyMgen.d(2,2, rawData=ch2pn(c("2","2*x+x^2","x","-1")))

X  # signing a polyMatrix
-X
+X

X+1  # polyMatrix + numeric
X-1

1+X  # numeric + polyMatrix
1-X

(1-X)*2  # polyMatrix * numeric
(-2)*(1-X)

X  # polyMatrix + matrix
diag(2)*X+diag(2)
-diag(2)-X-diag(2)

X-M2pM(diag(2))  # polyMatrix - polyMatrix -- OK
X+Y
X-Y

# polyMatrix * polynom -- three solutions:
X*"1+3*x+2*x^2"  # as a string
"1+3*x+2*x^2"*X

X*diag(list(ch2pn("1+3*x+2*x^2")),dim(X)[2])  # as a diagonal polyMatrix
diag(list(ch2pn("1+3*x+2*x^2")),dim(X)[2])*X

# X*polynom::polynomial(c(1,3,2))  # as a polynomial
# polynom::polynomial(c(1,3,2))*X  # does not works!

# polyMatrix product
Z<-X*Y
Z
const(X)
const(Y)
const(X)*%*%const(Y)  # conventional multiplication of two matrices
const(Z)  # similar: the constant of the product polyMatrix
const(X)*const(Y) # dissimilar: element wise product of two conventional matrices

# polyMatrix power
X
X^0
X^1
X^2
X*X
X^3

# clean up
# rm(X, Y, Z)

---

**pMadj**

*The adjugate of the given polynomial matrix*

**Description**

The adjugate of the given matrix. Also the transpose of the matrix formed from the determinants of submatrices multiplied by the chessboard rule signs.

**Usage**

`pMadj(pm)`

**Arguments**

- `pm` a `polyMatrix` class object

**Value**

A `polyMatrix` class object, the adjugate of the given polynomial matrix.

**See Also**

`pMdet`, `pMsub`

**Examples**

```
(A<-polyMgen.a(2,2))
# 1 + 2*x + 3*x^2    7 + 8*x + 9*x^2 + 10*x^3
# 4 + 5*x + 6*x^2    11 + 12*x
pMdet(A)
# -17 - 33*x - 61*x^2 - 97*x^3 - 104*x^4 - 60*x^5
pMsub(A,1,1)*pMsub(A,2,2)-pMsub(A,1,2)*pMsub(A,2,1)
# -17 - 33*x - 61*x^2 - 97*x^3 - 104*x^4 - 60*x^5
(B<-pMadj(A))
# 11 + 12*x          -7 - 8*x - 9*x^2 - 10*x^3
# -4 - 5*x - 6*x^2    1 + 2*x + 3*x^2
A*B                   # diagonal
```
### pMbas

#### Bastion elements of a polynomial matrix

**Description**

Elements of a matrix or polyMatrix object, from non-beating positions for bastion in the matrix as a chess board.

**Usage**

\[
\text{pMbas}(\text{pm}, \text{ki}, \text{byrow})
\]

**Arguments**

- **pm**: a polynomial matrix object
- **ki**: a permutation of the column numbers
- **byrow**: logical. If FALSE (the default) the elements of ki are column indexes, if TRUE these are row indexes.

**Details**

Usual
pMcol

**Value**

A list of polynomial class bastion elements of the given `polyMatrix` determined by the given permutation `ki` in a `pVector` object.

**See Also**

`pMdet`

**Examples**

```r
A <- polyMgen.a()
class(A)# "polyMarray" "polyMatrix"
dim(A)# 2 x 3
A
pMbas(A, c(2,3), byrow=TRUE) # the [1,2] and [2,3] elements
pMbas(A, c(2,2,1), byrow=FALSE) # the [2,1], [2,2] and [1,3] elements
pMbas(A, c(2,2), byrow=FALSE) # the [2,1] and [2,2] elements
# pMbas(A, c(2,2,1), byrow=TRUE) # Error: Index vector too long!

A <- polyMgen.d(3,3,
   rawData=ch2pm(c("1","0","-1",
    "2","1","-1",
    "0","1","1"), byrow=TRUE)
   A
pMbas(A, c(2,3,1), byrow=TRUE) # the [1,2], [2,3] and [3,1] elements
pMbas(A, c(2,3,1), byrow=FALSE) # the [2,1], [3,2] and [1,3] elements

# clean up
# rm(A)
```

---

**pMcol**

A column of the given `polyMatrix`

**Description**

Picks up a given column of the given matrix.

**Usage**

`pMcol(pm, which = 1)`

**Arguments**

- `pm` a `polyMatrix` class object
- `which` the serial number of the required column

**Value**

A `pVector` class list of the elements of the column 'which' of the polynomial matrix 'pm'.
See Also

  pMrow

Examples

A <- polyMgen.a()
A

pMcol(A, 2)

# clean up
# rm(A)


pMdet

Calculate the determinant of a real or polynomial matrix

Description

The determinant of the given square matrix.

Usage

pMdet(pm)

Arguments

  pm  a polyMatrix class object

Value

  A numeric or polynomial class polynom.

See Also

  pMsub, pMadj

Examples

rd <- c(3, 0:7)
D <- polyMgen.a(3, 3, rawData=rd, degree=1, symb="x")
D
# 3 5 + 6*x 2 + 3*x
# 1 + 2*x 7 + 3*x 4 + 5*x
# 3 + 4*x x 6 + 7*x

pMdet(D) # 114 + 150*x + 22*x^2 + 6*x^3
elem <- function(x,k,j) (pMbas(x,c(rep(1,k-1),j),byrow=TRUE)$dlist[[k]][[1]])
d11 <- elem(D,1,1);d12 <- elem(D,1,2);d13 <- elem(D,1,3)
d21 <- elem(D,2,1);d22 <- elem(D,2,2);d23 <- elem(D,2,3)
d31 <- elem(D,3,1);d32 <- elem(D,3,2);d33 <- elem(D,3,3)

polyMgen.d(3, 3, rawData=list(d11,d12,d13,d21,d22,d23,d31,d32,d33), symb="x", byrow=TRUE)

# direct det calculation
d11*d22*d33+d12*d23*d31+d13*d21*d32-(d13*d22*d31+d11*d23*d32+d12*d21*d33)
# 114 + 150*x + 22*x^2 + 6*x^3

D <- polyMgen.a(3,3,rawData=rd,degree=0)
D

M <- matrix(rd, ncol = 3)
M

c(pMdet(D),det(M))  # det of a polyMatrix == det of a real matrix (!!)

# clean up
# rm(D, d11, d12, d13, d21, d22, d23, d31, d32, d33, elem, M, rd)

---

**pMdiag**

Generates a diagonal polynomial square matrix

---

**Description**

The generated matrix a diagonal matrix of the given dimension with the diagonal elements given as input.

**Usage**

pMdiag(p, diag_dim, symb="x")

**Arguments**

- **p**  
  A polynomial object or list of polynomial objects for diagonal elements
- **diag_dim**  
  The row and column size of the diagonal matrix
- **symb**  
  The symbol used in the polynomial matrix

**Value**

A `polyMatrix` class type polynomial matrix with given diagonal elements.

**See Also**

diag
Examples

\[
\text{pMdiag(ch2pn("1"),3)}
\]
\[
\text{pMdiag(ch2pn("1+2*x+3*x^2"),3)}
\]
\[
\text{pMdiag(ch2pn(c("x","2*x","3*x","4*x")),3)}
\]
\[
\text{pMdiag(ch2pn(c("x","2*x","3*x","4*x")),6)}
\]

Description

Calculates the Kronecker indices of a polynomial matrix.

Usage

\[
\text{pMkmm(pm)}
\]

Arguments

\[
\text{pm} \quad \text{a polynomial matrix}
\]

Value

A vector of the Kronecker indices.

Note

The method doesn’t work yet.

See Also

polyMatrix-package

Examples

\[
\text{polyMgen.a()}
\]

#% end of donttest
Description

The product of the elements of the vector `pm` given in `polyMatrix` class object format.

Usage

```r
pMprod(pm)
```

Arguments

- `pm`  
  a `polyMatrix` class object

Details

The input must be a `polyMatrix` class vector object. The result is a `polynomial` class `polynom`.

Value

A `polynomial` which is the product of the elements of the argument `pm`.

See Also

- `pVsk`

Examples

```r
set.seed(32)
A <- polyMgen.a(1,2)
A
pMprod(A)

# clean up
# rm(A)
```
### pMrow

*A row of the given polyMatrix*

**Description**

Picks up a given row of the given polyMatrix.

**Usage**

```

pmrow(pm, which = 1)
```

**Arguments**

- `pm`: a `polyMatrix` class object
- `which`: the serial number of the required column

**Value**

A `pVector` class list of the elements of the `which` row of the `pm` polynomial matrix.

**See Also**

- `pmrow`

**Examples**

```
A <- polyMgen.a()
A
pmrow(A, 2)

# clean up
# rm(A)
```

### pMsgn

*Change the sign of a polynomial matrix object*

**Description**

Multiplies the given polyMatrix by -1, and returns the -1 times the given polynomial matrix.

**Usage**

```

pMsgn(pm)
```

**Arguments**

- `pm`: a polynomial matrix object
Value

The polynomial matrix for which the \( pm + pMsgn(pm) \) equals a zero polynomial matrix.

See Also

Ops.polyMatrix

Examples

```r
A <- polyMgen.a()
A
class(A) # "polyMarray" "polyMatrix"

A - polyMconvert(A,"polyMbroad")
A - polyMconvert(A,"polyMcells")
A - polyMconvert(A,"polyMdlist")

A + pMgsn(polyMconvert(A,"polyMbroad"))
A + pMgsn(polyMconvert(A,"polyMcells"))
A + pMgsn(polyMconvert(A,"polyMdlist"))

# clean up
# rm(A)
```

Description

Checks the consistency of a polyMatrix object. In the polyMatrix there are four different but equivalent storing methods of polynomial matrices.

Usage

`pMstr(pm)`

Arguments

`pm` a polynomial matrix

Value

A logical value of the consistency of the given polynomial object. In case of inconsistency an information about the errors.

See Also

`polyMgen.a, polyMgen.b, polyMgen.c, polyMgen.d`
Examples

```r
A <- polyMgen.a(2,3,5)
B <- polyMconvert(A,"polyMdlist")

pMstr(B)$cns

## class of elements: polynomial
B[[1]][[1]]<-1
str(B)
(pMstr(B))

## class of sublists: list
B[[1]]<-1:length(B[[1]])
str(B)
pMstr(B)

# absence of an element
B[[1]]<-as.list(B[[1]][-1])
str(B)
pMstr(B)[[3]]

# clean up
# rm(A, B)

# % end of donttest
```

---

**pMsub**  
*A submatrix of a polynomial matrix*

**Description**

Retains or deletes the i-th rows and the j-th column, depending of the sign of these indices.

**Usage**

```r
pMsub(pm, i, j)
```

**Arguments**

- `pm`  
a polynomial matrix
- `i`  
the number of rows to retain or delete
- `j`  
the number of columns to retain or delete

**Details**

If the row or column number is NULL, that means all elements from the given columns or rows respectively. If `j` is not given, that means a symmetric submatrix.
Value

A polynomial matrix which is a submatrix of the given matrix in ‘polyMdlist’ format.

See Also

pMdet, pMadj

Examples

A <- polyMgen.a(3, 4, 1:24, degree=1)
A
pMsub(A, 2, 3)
pMsub(A, 1, NULL)
pMsub(A, NULL, 3)
pMsub(A, 1:2, 2:4)
pMsub(A, -3, 2:5)
pMsub(A, 1:2)

# clean up
# rm(A)

pn2ch

Converts a polynom class object to a character class object

Description

Takes a polynom class object and returns a character class object looks like the print image of the polynom object.

Usage

pn2ch(x, symb = "x", digits = getOption("digits"), decreasing = FALSE, ...)

Arguments

x the given polynom class object
symb the symbol of polynom
digits the number of significant digits to be printed
decreasing the desired order of the terms of the polynomial
... additional arguments

Details

A technical subrutin to convert a polynom object to a character object.

Value

A character class object.
polyMconvert

See Also

pprt

Examples

p <- polynom::polynomial(0:12)
pn2ch(p)

p <- polynom::polynomial(12:0)
pn2ch(p)

# clean up
# rm(p)

polyMconvert  Conversion between the pairs of the four polyMatrix subclasses

Description

The polyMatrix package has 4 different but equivalent methods to store the polynomial matrices. This function provides a conversion between the 4 methods. In the background operates one of the existing 12 subroutines, corresponding to the sub-class of the given input object and the required sub-class of the output object.

Usage

polyMconvert(pm, newclass)

Arguments

pm an arbitrary polyMatrix class object
newclass a class identifier character string, one of the following: "polyMarray", "polyMbroad", "polyMcells" or "polyMdlist"

Value

A polyMatrix class object, equivalent with the pm input object, but stored in the new, given format by the newclass parameter.

See Also

polyMatrix-package
### Examples

```r
set.seed(2)
pa <- polyMgen.a(rand=function(x) rpois(x,1)) # Poisson(1) distributed coefficients
class(pa) # "polyMarray" "polyMatrix"
str(pa)
pa

pb <- polyMconvert(pa, "polyMbroad")
class(pb) # "polyMbroad" "polyMatrix"
str(pb)
pb

pc <- polyMconvert(pa, "polyMcells")
class(pc) # "polyMcells" "polyMatrix"
str(pc)
pc

pd <- polyMconvert(pa, "polyMcells")
class(pd) # "polyMdlist" "polyMatrix"
str(pd)
pd

px <- polyMconvert(pa, "polyMarray")
class(px) # "polyMarray" "polyMatrix"
str(px)
px

# clean up
# rm(pa, pb, pc, pd, px)
```

---

### polyMgen

*Generate a polyMatrix class polynomial matrix*

### Description

A complex tool to generate a polynomial matrix in 'polyMatrix' form. The 'polyMatrix' object contains the following three invariant elements: the $\text{dim}$, $\text{degree}$ and $\text{symb}$ independently of the sub-class of the object. The $\text{dim}$ is the size of the matrix. The $\text{degree}$ is a matrix of non-negative numbers, the degree of the polynomial elements of the polynomial matrix. The $\text{symb}$ the symbol to print and identify the reference to the algebraic set over which the polynom defined – sorry, this option not fully works.

### Usage

`polyMgen(nrow, ncol, rawData, symb, rand, degree, byrow, sm="polyMarray", ...)`
**Arguments**

- **nrow** the desired number of rows
- **ncol** the desired number of columns
- **degree** the desired degrees of polynomials when it is filled randomly
- **rawData** the data to fill with the polynomial matrix. In case of polyMgen.a, polyMgen.b and polyMgen.c a vector of the required coefficients of the polynomials. The case of polyMgen.d is different: here the rawData is a list of polynomial class objects
- **symb** the desired polynom symbol
- **rand** the name of the random generator used to generate the coefficients of a random polynomial matrix
- **byrow** logical. If FALSE (the default) the matrix is filled by columns, if TRUE the matrix is filled by rows
- **sm** storage method: polyMarray (the default) or\polyMbroad, polyMcells, polyMdlist
- ... additional arguments

**Details**

The four possible methods to generate the polynomial matrix from the given material rawData or (the default) or random values:

- ‘array’, when the coefficient matrices of the polinomial matrix are stored in two independent element of the list of the polyMatrix: the constant matrix in $const$, and the coefficients of the first, second... degree of the polynomial in the first, second,... element of the $array$. In this case the sub-class of the generated polyMatrix object is polyMarray.

- ‘broad-matrix’, when the coefficients are stored in one broad matrix in the $broad$ element of the polyMatrix object. The size of this broad matrix is: \(nrow \times (ncol \times (d+1))\), when the degree of the \(nrow \times ncol\) size polynomial matrix is \(d\). In this case the sub-class of the generated polyMatrix object is polyMbroad.

- ‘list of cells’, when the \(nrow \times ncol\) size coefficient matrices are stored in \(d+1\) element of the $cells$ list element of the polyMatrix object. In this case the sub-class of the generated polyMatrix object is polyMcells.

- ‘double list’, when the elements of the polynomial matrix are stored in the form of polynomial class objects in a \(nrow \times ncol\) element double list. In this case the sub-class of the generated polyMatrix object is polyMdlist.

**Value**

A polyMatrix class object with a sub-class polyMarray, polyMbroad, polyMcells or polyMdlist.

**See Also**

polyMatrix-package, polyMgen.a, polyMgen.b, polyMgen.c, polyMgen.d
Examples

polyMgen.a() # default size & coefficients & rank
polyMgen.a(2,3,1:6) # given coefficients & default rank
polyMgen.a(2,3,1:6,degree=1) # given coefficients & rank
polyMgen.a(2,3,1:36,degree=matrix(0:5,2,3)) # given coefficients & ranks
polyMgen.a(2,3,1:12,degree=matrix(2:0,2,3)) # given coefficients & ranks
polyMgen.a(rand=TRUE) # normally distributed coefficients
polyMgen.a(rand=rexp) # exponentially distributed coefficients

pois.vg.fv <- function(x) rpois(x,1)
polyMgen.a(rand=pois.vg.fv) # Poisson(1) distributed coefficients
rm(pois.vg.fv)

polyMgen.b() # class="polyMbroad", broad matrix form
polyMgen.c() # class="polyMcells", list of coefficient matrices
polyMgen.d() # class="polyMdlist", dubble list

polyMgen.d(2,2,
  rawData=list(polynom::polynomial(1),
               polynom::polynomial(c(0,0,1)),
               polynom::polynomial(c(0,1)),
               polynom::polynomial(0)))

px <- polyMgen.d(3,3,rawData=ch2pn(c(-3 + z^2,"2 + 4*z","-z^2",
                       "1","2","3 + z",
                       "2*z","0","2 - 3*z"),"z"))
class(px) # "polyMdlist" "polyMatrix"
px

px <- polyMgen.d(3,3,rawData=ch2pn(c(-3 + z^2,"2 + 4*z","-z^2",
                       "1","2","3 + z",
                       "2*z","0","2 - 3*z"),"z"),byrow=TRUE)
class(px) # "polyMdlist" "polyMatrix"
px

polyMgen.d()
polyMgen.d(rand=TRUE)
polyMgen.d(rand=TRUE,degree=2)
polyMgen.d(degree=3)
polyMgen.d(degree=NULL)

# clean up
# rm(px)
polyMul  

Multiplication of two polynomial by matrix multiplication

Description

...

Usage

polyMul(p, q)

Arguments

p ... polyMgen
q ... polyMgen

Details

more details than the description above

Value

Describe the value returned!! If it is a LIST, use

comp1 Description of 'comp1'

Note

further notes

Author(s)

who you are

References

references to the literature

See Also

objects to See Also as polyMatrix-package
Examples

```r
set.seed(12345)
p <- polynom::polynomial(rpois(rgeom(1,.2)+1,2))
q <- polynom::polynomial(rpois(rgeom(1,.2)+1,2))
p
q
p * q
as.numeric(p * q) # 9 30 30 18 19 14 7 3
polyMul(p, q)
polyMul(q, p)

# clean up
# rm(p, q)
```

---

**pprt**

*Intelligent print of a polynom object*

**Description**

Calls the pn2ch converter, and prints the polynomial considering shift parameter and the width of the actual console window.

**Usage**

```r
pprt(x, symb = "x", shift = 0, digits = getOption("digits"), decreasing = FALSE, ...)
```

**Arguments**

- `x`: the polynom object to be printed
- `symb`: the symbol used in the print
- `shift`: the beginning shift
- `digits`: the width of the coefficients to be printed
- `decreasing`: indicator of the order of powers terms: decreasing or not
- `...`: additional arguments

**See Also**

- `pn2ch`

**Examples**

```r
p <- polynom::polynomial(0:12)
pprt(p)
pprt(p, sh=3)
pprt(p, sh=3, symb="L")
# clean up
# rm(p)
```
predict.charpn

Calculate the value of a characteristic polynomial for a polyMatrix argument

Description

The `polynom::predict.polynomial()` method works by real coefficients only but it works for real or complex and matrix argument also. But in the case of matrix argument uses the elementwise product. The `polyMatrix::predict` method evaluate the polynom by the usual matrix product definition.

Usage

```r
## S3 method for class 'charpn'
predict(object, pM, ...)  
```

Arguments

- `object`: a `charpn` class object
- `pM`: a `polyMatrix` class object
- `...`: additional arguments

Value

- `value`: A `polynomial` class object

See Also

`predict.polynomial`, `predict.polyMatrix`

Examples

```r
pm <- polyMgen.d(2, 2, rawData = ch2pn(c("1", "x^2", "x", "0")))
pm # 1, z^2\ z, 0

predict(pm, 2) # matrix(c(1, 4, 2, 0), 2)

# predict the result of a linear model
x <- 1:5; y <- rnorm(5); predict(lm(y ~ x)) # the used method: predict.lm()

# clean up
# rm(pm, x, y)
```
predict.polyMatrix

Calculate the value of a polynomial matrix for real or complex values

Description

Calculate the value of the polynom elements of the ‘polyMatrix’ class object for real and complex argument also.

Usage

## S3 method for class 'polyMatrix'
predict(object, M, ...)

Arguments

- **object**: a ‘polyMatrix’ class object
- **M**: a real or complex argument
- **...**: additional arguments

Value

A matrix the just the same size as the input, filled by the values of the polynom elements, at the given argument.

See Also

predict.polynomial, predict.charpn

Examples

# for predict the value of a polynomial matrix at a real or complex value
# the used method: predict.polyMatrix()

pm <- polyMgen.d(2,2, ch2pn(c("1","x","x^2","0")))
pm # 1, x^2 \ x, 0
predict(pm, 2) # 1, 4 \ 2, 0

# for predict the result of a linear model
# the used method: predict.lm()

x <- 1:5; y <- rnorm(5); predict(lm(y~x))

# clean up
# rm(pm, x, y)
**predict.polynomial**

Calculate the value of a ‘polynomial’ for real, complex and matrix argument

### Description

The `polynom::predict.polynomial()` method works for real, complex and matrix argument also. But in the case of matrix argument it uses the elementwise calculations instead the matrix multiplication. The `polyMatrix::predict.polynomial()` calculate the result by the usual matrix product definition except the case, when the parameter `meth` equals by "as.in.the.polynom.package".

### Usage

```r
## S3 method for class 'polynomial'
predict(object, M, meth = c("as.matrix","as.in.the.polynom.package"),...)
```

### Arguments

- `object` a numeric, matrix or polyMatrix class object
- `M` a polynomial class object
- `meth` If the value of this parameter as the default, "as.matrix" then calculate by matrix multiplications. Otherwise it calculate by elementwise multiplications.
- `...` additional arguments

### Value

The is a numeric, numeric, matrix or polyMatrix class object depending on the class of the input

### See Also

`predict.polyMatrix`, `predict.charpn`

### Examples

```r
p <- polynom::polynomial(1:3)
p # 1 + 2*x + 3*x^2

# predict a polynom for real values
predict(p, 1) # 6

# predict a polynom for complex values
predict(p, 1i) # -2+2i

# predict a polynom for matrices
M <- matrix(c(1,-1),2,2);
M
predict(p, M) # 3,2 \ -2, -1
```
# mimicking the elementwise calculation of the masked "polynom::predict" method
predict(p,M,meth="as.in.") # 6, 6 \ 2, 2

# predict a polynom for a polynomial matrix
pM <- polyMgen.d(2,2, rawData=ch2pn(c("1","x^2","x","0")))
P # 1, x \ x^2, 0
predict(p,pM) # method: predict.polyMatrix()
# 6 + 3*x^3 5*x
# 5*x^2 1 + 3*x^3

# ---
# predict the result of a linear model
x <- c(1,3,4,1,0);
y <- c(27,81,54,27,18)
predict(lm(y~x)) # method: predict.lm()
# 31 57 70 31 18

# clean up
# rm(p, M, pM, x, y)

---

**print**

*Prints a polynomial matrix or a varma model in the given form*

**Description**

Printing method for class polyMatrix and pMvarma.

**Usage**

```r
## S3 method for class 'polyMatrix'
print(
  x, style=c("matrix", "polynom", "broad", "raw"),
  round=NULL, digits=getOption("digits"), shift = 3, decreasing = FALSE, ...
)
## S3 method for class 'pMvarma'
print(
  x, style=c("matrix", "polynom", "broad", "raw"),
  round=NULL, digits=getOption("digits"), shift = 3, ...
)
```

**Arguments**

- `x`: a polyMatrix or pMvarma class object to be printed
- `style`: one of the four printing types
- `round`: rounds as the base::round() function
- `digits`: the number of significant digits to be printed
- `shift`: the beginnig shift
- `decreasing`: the terms of the polynomial in decreasing order or not
- `...`: additional arguments
See Also

print

Examples

set.seed(1)
pm <- polyMgen.a(rand=function(x) rpois(x,1))

pa <- polyMconvert(pm,"polyMarray") # class: "polyMarray" "polyMatrix"
pb <- polyMconvert(pm,"polyMbroad") # class: "polyMbroad" "polyMatrix"
pc <- polyMconvert(pm,"polyMcells") # class: "polyMcells" "polyMatrix"
pd <- polyMconvert(pm,"polyMdlist") # class: "polyMdlist" "polyMatrix"
# clean up
# rm(pm)

# the default, the "matrix" print image style: matrix of polynomials
pa
pb
pc
pd

# the three other print image style:
print(pa,"poly")
print(pa,"broad")
print(pa,"raw")

print(pb,"poly")
print(pb,"broad")
print(pb,"raw")

print(pc,"poly")
print(pc,"broad")
print(pc,"raw")

print(pd,"poly")
print(pd,"broad")
print(pd,"raw")

# clean up
# rm(pa,pb,pc,pd)

varma <- polyMgen.varma()

varma
print(varma,"matrix") # the same print image
print(varma,"poly")
print(varma,"broad")
print(varma,"raw")

# clean up
# rm(varma)
Description
The program returns a logical value, whether the given polynomial matrix is column or row proper or both. A ‘proper’ matrix called in other way column or row ‘reduced’.

Usage
proper(pm,type=c("col","row","both"),print=c(TRUE,FALSE))

Arguments
pm a polyMatrix class polynomial matrix
type the type of rating row//col//both
print the kind of printing

Details
The program calculates first the column or row associated matrices, and the answer depends on there ranks. A polynomial matrix is ‘proper’ if the associated matrix has a full rank. The program has an invisible silent output.

Value
Logical: TRUE or FALSE. The associated matrix is only an optional part of the output, and the print.

See Also
coefs, lead

Examples
pm <- polyMgen.d(2,2,rawData=ch2pn(c("-1+7*x","x","3-x+x^2","-1+x^2-3*x^3")))

# proper(pm) # the default is the col-property labeling
(proper(pm,print=FALSE)) # the invisible output, without printing
proper(pm,"col")
proper(pm,"row")
proper(pm,"both")

# clean up
# rm(pm)
The scalar product of two polynomial vectors

Description

If `pMy` is `NULL`, then the result is the sum of squares of the elements of `pMx`. Otherwise equals the scalar product of the two given polynomial vectors.

Usage

`pVsk(pMx, pMy=NULL)`

Arguments

- `pMx`: a `polyMatrix` class row or column vector
- `pMy`: a `polyMatrix` class row or column vector

Details

The two vectors must have the same length.

Value

A polynomial class object, which is the sum of the elementwise product of the two vectors.

See Also

`pMprod`

Examples

```R
A <- polyMgen.d(2,2,rawData=ch2pn(c("-3","2+4*x","-x^2","1"),'z'))
A

pMcol(A,1) # "-3", "2 + 4*x"
pVsk(pMcol(A,1)) # "13 + 16*x + 16*x^2"

pMrow(A,2) # "2 + 4*x", "1"
pVsk(pMcol(A,1),pMrow(A,2)) # "-4 - 8*x"

# clean up
# rm(A)
```
symb

The printing symbol of the given polynomial matrix

Description
Get the printing symbol of the given polynomial matrix

Usage
symb(pm)

Arguments
pm
a polyMatrix class object

Details
The polynomial matrix objects contains a character to sign the variable of the polynomials. The system default is 'x'.

Value
One character, which used to print the polynomials in default cases

See Also
dim.polyMatrix, degree.polyMatrix, degree.polynomial

Examples

pM <- polyMgen.d(2,2,rawData=
    ch2pn(c("-3+s^3","2+4*s","s^2","1"),symb="s"))
pM
# -3 + x^3  x^2
# 2 + 4*x  1
symb(pM) # "x"

pM <- polyMgen.d(2,2,rawData=
    ch2pn(c("-3+x^3","2+4*x","x^2","1"),symb="L"))
pM
# -3 + L^3  L^2
# 2 + 4*L  1
symb(pM) # "L"

# clean up
# rm(pM)
Matrix transpose of a 'polyMatrix' class object

Description
Given a polyMatrix class \( x \), the call \( t(x) \) returns the transpose of \( x \).

Usage
## S3 method for class 'polyMatrix'
t(x, ...)

Arguments
  x  an polyMatrix class object
  ... not used

Details
The method first convert the storage method of the given polyMatrix object to polyMdlist class interpretation, then flips the polyMatrix over its diagonal.

Value
A 'polyMdlist, polyMatrix' class object, the transposed version of the given \( x \) polynomial matrix.

See Also
The ‘t’ in the base package.

Examples
# ---
# case of real matrix argument
m <- matrix(1:12,3,4)
m
  
t(m) # the \code{base::t()} function
# ---
# case of polynomial matrix argument
pa <- polyMgen.a()  
pa
  
dim(pa) # 2 x 3

pm <- t(pa)
  
dim(pm) # 3 x 2
Description

Returns the trace of the given matrix.

Usage

tr(matrixObject)

Arguments

matrixObject  a matrix or polyMatrix class object

Details

Calculates the sum of the diagonal elements of the given matrix.

Value

• a numeric class object, if the given matrix is a matrix class object
• a polynomial class object, if the given matrix is a polyMatrix class object

See Also

tr.polyMatrix, polyMatrix-package, polynomial, but the trace is a debugging utility

Examples

# the case of a matrix class input object
M <- matrix(1:9,3,3)
class(M)
M
# 1 4 7
# 2 5 8
# 3 6 9
tr(M) # 15

(M <- matrix(1:12,3,4))
# 1 4 7 10
# 2 5 8 11
# 3 6 9 12
triang_Euclidean

**Triangularization of a polynomial matrix by Euclidean division method**

**Description**

The function `triang_Euclidean` triangularize the given polynomial matrix.

**Usage**

`triang_Euclidean(pm)`

**Arguments**

- `pm` polynomial matrix to triangularize

**Details**

The method use the for polynomials extended Euclidean algorithm.

This method search a solution of the triangulrization by the method of Sylvester matrix, descripted in the article Labhalla-Lombardi-Marlin (1996).

**Author(s)**

Nikolai Ryzhkov, <namezys@gmail.com>
triang_Sylvester

References


See Also

triang_Sylvester

triang_Sylvester Triangularization of a polynomial matrix by Sylvester method

Description

The function triang_Sylvester triangularize the given polynomial matrix.

The u parameter is a necessary supplementary input without default value. This parameter give the minimal degree of the searched triangulizator to solve the problem.

Usage

triang_Sylvester(pm, u, eps=ZERO_EPS)

Arguments

pm polynomial matrix to triangularize
u the minimal degree of the triangularizator multiplicator
eps toleranz limit

Details

In a polynomial matrix the head elements are the first non-zero polynomials of columns. The sequence of row indices of this head elements form the shape of the polynomial matrix. A polynomial matrix is in left-lower triangular form, if this sequence is monoton increasing.

This method search a solution of the trianglerization by the method of Sylvester matrix, descripted in the article Labhalla-Lombardi-Marlin (1996).

Value

T the left-lower triangularized version of the given polynomial matrix
U the right multiplicator to triangularize the given polynomial matrix

Author(s)

Nikolai Ryzhkov, <namezys@gmail.com>

References

Description

Matrix multiplication for polyMatrix class polynomial matrices. Multiplies the two matrices only if they are conformable.

Usage

left %XX% right

Arguments

left a polyMatrix class object, scalar number or polynomial
right a polyMatrix class object, scalar number or polynomial

Details

The number of columns in left must be equal to number of rows in right.

Value

A polyMatrix class object which is the product of the polyMatrix class left and right matrices.

See Also

polyMatrix-package

Examples

```r
pm <- polyMgen.a()
pm * t(pm)

# two constante matrices
a <- matrix(1:12,3,4)
b <- matrix(1:20,4,5)
a %XX% b

a <- M2pM(a)
b <- M2pM(b)
a * b
a %XX% b

pM2M(a) %XX% pM2M(b)
```
# clean up
# rm(pm, a, b)
## Index

### arith
- predict.charpn, 42
- predict.polyMatrix, 43
- predict.polynomial, 44

### check
- is.polyMatrix, 19

### datagen
- pMdiag, 29
- polyMgen, 37

### manip
- CommonPolynom, 10
- pMbas, 26
- pMcol, 27
- pMrow, 32
- pMstr, 33
- pMsub, 34

### math
- %X%, 54
- pMadj, 25
- pMdet, 28
- pMkmn, 30
- pMprod, 31
- polyMul, 40
- pVsk, 48

### package
- polyMatrix-package, 3

### print
- pprt, 41
- print, 45

### symbolmath
- Ops.polyMatrix, 23

### triangularization
- triang_Euclidean, 52
- triang_Sylvester, 53

### utilities
- CanForm, 4
- ch2pn, 5
- charpn, 7
- coefs, 9
- const, 12
- constConv, 13
- cycFill, 14
- degree, 15
- lead, 20
- matrixMinMax, 21
- MTS2pM, 22
- pn2ch, 35
- polyMconvert, 36
- proper, 47
- symb, 49
- tr, 51
- %X%, 54

CanForm, 4
ch2pn, 5
charpn, 7
coeifs, 9, 12, 13, 21, 47
coeifs.polyMatrix, 10
coeifs.polynomial, 10
colMax (matrixMinMax), 21
colMin, 22
colMin (matrixMinMax), 21
CommonPolynom, 10
const, 10, 12, 21
const.polyMatrix, 12
const.polynomial, 12
constConv, 13
cycFill, 14
degree, 13, 15, 21
degree.polyMatrix, 15, 49
degree.polynomial, 15, 49
diag, 16, 29
dim, 18
dim.polyMatrix, 49
GCD (CommonPolynom), 10
is.polyMatrix, 19
<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCM (CommonPolynom)</td>
<td>10</td>
</tr>
<tr>
<td>lead</td>
<td>10, 12, 13, 15, 20, 47</td>
</tr>
<tr>
<td>M2pM (constConv)</td>
<td>13</td>
</tr>
<tr>
<td>matrixMinMax</td>
<td>21</td>
</tr>
<tr>
<td>MTS2pM</td>
<td>22</td>
</tr>
<tr>
<td>Ops</td>
<td>24</td>
</tr>
<tr>
<td>Ops.polyMatrix</td>
<td>23, 33</td>
</tr>
<tr>
<td>pM2M (constConv)</td>
<td>13</td>
</tr>
<tr>
<td>pMadj</td>
<td>11, 25, 28, 35</td>
</tr>
<tr>
<td>pMbas</td>
<td>26</td>
</tr>
<tr>
<td>pMcol</td>
<td>27</td>
</tr>
<tr>
<td>pMdet</td>
<td>11, 25, 27, 28, 35</td>
</tr>
<tr>
<td>pMdiag</td>
<td>29</td>
</tr>
<tr>
<td>pMkmm</td>
<td>30</td>
</tr>
<tr>
<td>pMprod</td>
<td>31, 48</td>
</tr>
<tr>
<td>pMrow</td>
<td>28, 32, 32</td>
</tr>
<tr>
<td>pMsgn</td>
<td>32</td>
</tr>
<tr>
<td>pMsub</td>
<td>25, 28, 34</td>
</tr>
<tr>
<td>pn2ch</td>
<td>5, 35, 41</td>
</tr>
<tr>
<td>polyMatrix-package</td>
<td>3</td>
</tr>
<tr>
<td>polyMconvert</td>
<td>36</td>
</tr>
<tr>
<td>polyMgen</td>
<td>37</td>
</tr>
<tr>
<td>polyMgen.a</td>
<td>14, 33, 38</td>
</tr>
<tr>
<td>polyMgen.b</td>
<td>33, 38</td>
</tr>
<tr>
<td>polyMgen.c</td>
<td>33, 38</td>
</tr>
<tr>
<td>polyMgen.d</td>
<td>14, 33, 38</td>
</tr>
<tr>
<td>polyMul</td>
<td>40</td>
</tr>
<tr>
<td>polynomial</td>
<td>20, 38, 51</td>
</tr>
<tr>
<td>pprt</td>
<td>5, 36, 41</td>
</tr>
<tr>
<td>predict.charpn</td>
<td>7, 42, 43, 44</td>
</tr>
<tr>
<td>predict.polyMatrix</td>
<td>42, 43, 44</td>
</tr>
<tr>
<td>predict.polynomial</td>
<td>42, 43, 44</td>
</tr>
<tr>
<td>print</td>
<td>45, 46</td>
</tr>
<tr>
<td>print.pMvarma</td>
<td>23</td>
</tr>
<tr>
<td>print.polyMatrix</td>
<td>23</td>
</tr>
<tr>
<td>proper</td>
<td>47</td>
</tr>
<tr>
<td>pVsk</td>
<td>31, 48</td>
</tr>
<tr>
<td>rowMax</td>
<td>22</td>
</tr>
<tr>
<td>rowMax (matrixMinMax)</td>
<td>21</td>
</tr>
<tr>
<td>rowMin</td>
<td>22</td>
</tr>
<tr>
<td>rowMin (matrixMinMax)</td>
<td>21</td>
</tr>
<tr>
<td>symb</td>
<td>49</td>
</tr>
<tr>
<td>t</td>
<td>50</td>
</tr>
<tr>
<td>tr</td>
<td>51</td>
</tr>
<tr>
<td>tr.polyMatrix</td>
<td>51</td>
</tr>
<tr>
<td>trace</td>
<td>51</td>
</tr>
<tr>
<td>triang_Euclidean</td>
<td>52, 54</td>
</tr>
<tr>
<td>triang_Sylvester</td>
<td>53, 53</td>
</tr>
</tbody>
</table>