Package: polyMatrix (via r-universe)

August 31, 2024

Version 0.9.11

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 3 methods of triangularization of polynomial matrices: Extended Euclidean algorithm which is most classical but numerically unstable; Sylvester algorithm based on LQ decomposition; Interpolation algorithm is based on LQ decomposition and Newton interpolation. 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>.

Type Package

Imports methods, polynom, Matrix

License MIT + file LICENSE

Depends R (>= 4.0)

Suggests testthat, withr

URL https://github.com/namezys/polymatrix

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

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Repository https://namezys.r-universe.dev

RemoteUrl https://github.com/namezys/polymatrix

RemoteRef HEAD

RemoteSha abdde4801218d043e6999a571c32542fae873983

Contents

`adjoint` *Adjugate or classical adjoint of a square matrix*

Description

The adjugate or classical adjoint of a square matrix is the transpose of its cofactor matrix. It is also occasionally known as adjunct matrix,[though this nomenclature appears to have decreased in usage.

Usage

```
adjoint(x)

## S4 method for signature 'polyMatrix'
adjoint(x)
```

Arguments

x	an matrix

Methods (by class)

- polyMatrix: adjugate of polynomial matrix

---

cbind *Combine polynial matrices by rows or coluns*

Description

Combine polynial matrices by rows or coluns

Usage

```
cbind(..., deparse.level = 1)

rbind(..., deparse.level = 1)
```

Arguments

...	(generalzed) vectors or mmatrices. If any of objects are polynomail matrix
deparse.level	details in base function, polynomial matrices doesn't use it

Value

if at least one argument is a polynomail matrix, the result will be combined polynomial matrix. Overwise, base package implementatioon base::cbind() or base::rbind() will be called.

Functions

- rbind: row based bind

See Also

base::cbind()

charpolynom *Characteristic polynomial of matrix*

Description

Characteristic polynomial of matrix

Usage

```
charpolynom(x)

## S4 method for signature 'matrix'
charpolynom(x)

## S4 method for signature 'polynomial'
charpolynom(x)

## S4 method for signature 'polyMatrix'
charpolynom(x)

## S4 method for signature 'polyMatrixCharPolynomial,ANY'
x[[i]]

## S4 method for signature 'polyMatrixCharPolynomial'
degree(x)

## S4 method for signature 'polyMatrixCharPolynomial'
predict(object, newdata)

## S4 method for signature 'polyMatrixCharPolynomial'
show(object)
```

Arguments

x	an matrix
i	the degree to extract polinomial coefficient
object	an R object
newdata	the value to evaluate

Details

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

Value

When the input is a numerical matrix of `matrix` class then the value is a `polynomial` object.

When the input is a `polyMatrix` object then a value is `polyMatrixCharClass` class object,

Methods (by class)

- `matrix`: for numerical matrix it is a polynomial with numerical coefficients
- `polynomial`: for polynomial it treats as a matrix 1x1
- `polyMatrix`: for polynomial matrix it has polynomial coefficients
- `x = polyMatrixCharPolynomial,i = ANY`: get polynomial coefficient of characteristic
- `polyMatrixCharPolynomial`: the degree of char polynomail of polynomial matrix
- `polyMatrixCharPolynomial`: value of char polynomail in polynomial point
- `polyMatrixCharPolynomial`: prints out a text representation of a characteristic polinomial of polinomial matrix

See Also

polyMatrixCharClass

Examples

```
# numerical matrices
m <- matrix(c(2, 1,
              -1, 0), 2, 2, byrow=TRUE)
charpolynom(m)
```

`cofactor` *Cofactor of matrix*

Description

Cofactor of matrix

Usage

```
cofactor(x, r, c)
```

Arguments

`x`	an matrix
`r, c`	the row and column

Value

cofactor which is number or polynomial

See Also

`adjoint()`

degree *Gets maximum degree of polynomial objects*

Description

Returns the maximum degree as an integer number.

Usage

```
degree(x)

## S4 method for signature 'numeric'
degree(x)

## S4 method for signature 'matrix'
degree(x)

## S4 method for signature 'polynomial'
degree(x)

## S4 method for signature 'polyMatrix'
degree(x)
```

Arguments

x	an R objects

Details

By default, this function raises error for unknown type of object.

A numerical scalar has zero degree.

A numerical matrix has zero degree as each of its items has zero degree as well.

For polynomials this function returns the highest degree of its terms with non-zero coefficient.

Value

The value is an integer number which can be different from zero only for polynomial objects.

Methods (by class)

- `numeric`: a scalar argument always has zero degree
- `matrix`: a numerical matrix always has zero degree
- `polynomial`: the degree of a polynomial
- `polyMatrix`: the degree of a polynomial matrix is the highest degree of its elements

Examples

```
# numerical
degree(1)  ## 0


# numerical matrix
degree(matrix(1:6, 3, 2)) ## 0


# polinomial
degree(parse.polynomial("1")) ## 0
degree(parse.polynomial("1 + x")) ## 1
degree(parse.polynomial("1 + x^3")) ## 3


# polynomial matrices
degree(parse.polyMatrix(
   "x; x^2 + 1",
   "0; 2x"))
## 2
```

`diag` *Polynomial matrix Diagonals Extract or construct a diagonal polynomial matrix.*

Description

Polynomial matrix Diagonals Extract or construct a diagonal polynomial matrix.

Usage

```
diag(x = 1, nrow, ncol, names = TRUE)

## S4 method for signature 'polynomial'
diag(x, nrow, ncol)

## S4 method for signature 'polyMatrix'
diag(x)
```

Arguments

`x`	a polynomial matrix, or a polynomial, or an R object
`nrow`, `ncol`	optional dimensions for the result when x is not a matrix.
`names`	not usedd

Details

In case of polynomail objets, `diag` has 2 distinct usage:

- `x` is a polynomial, it returns a polynomial matrix the given diagonal and zero off-diagonal entries.
- `x` is a polynomial matrix, it returns a vector as a polynomial matrix of diagonal elements

For polynomial, either `nrow` or `ncol` must be provided.

Methods (by class)

- `polynomial`: for a polynomial, returns polynomial matrix with given diagonal
- `polyMatrix`: for a polynomial matrix extract diagonal
  For polynomial matrix, neither `nrow` and `ncol` can't be provided.

See Also

Base `base::diag()` for numericals and numerical matrices

Examples

```
# numericals and numerical matrix
diag(matrix(1:12, 3, 4)) ## 1 5 8
diag(9, 2, 2)
##      [,1] [,2]
## [1,]    9    0
## [2,]    0    9



# polynomial
diag(parse.polynomial("1+x+3x^2"), 2, 3)
##                [,1]           [,2]   [,3]
## [1,]   1 + x + 3x^2              0      0
## [2,]              0   1 + x + 3x^2      0


# polynomial matrix
diag(parse.polyMatrix(
  "-3 + x^2, 2 + 4 x,  -x^2",
  "       1,       2, 3 + x",
  "      2x,       0, 2 - 3x"
))
##            [,1]   [,2]     [,3]
## [1,]   -3 + x^2      2   2 - 3x
```

GCD *GCD for polynomial matrices*

Description

The greatest common divisor of polynomials or polynomial matrices.

Usage

```
GCD(...)

## S4 method for signature 'polyMatrix'
GCD(...)
```

Arguments

...	an list of polynomial objects

Methods (by class)

- polyMatrix: the greatest common divisor of all elements of the polynomial matrice

See Also

polynomial implementation polynom::GCD() and LCM()

Examples

```
# GCD of polynomial matrix

GCD(parse.polyMatrix(
 "  1 - x, 1 - x^2, 1 + 2*x + x^2",
 "x - x^2,   1 + x, 1 - 2*x + x^2"
))  ## 1
```

inv *Inverse polynomial matrix*

Description

During inversion we will try to round to zero

Usage

```
inv(x, eps = ZERO_EPS)
```

Arguments

x	an polynomial matrix
eps	zero threshold

Details

Right now only matrices with numerical determinant is supported

---

is.polyMatrix *Check if object is polyMatrix*

Description

Check if object is polyMatrix

Usage

```
is.polyMatrix(x)
```

Arguments

x	an R object

Value

TRUE if object is a polonial matrix

Examples

```
is.polyMatrix(c(1, 2, 3))
is.polyMatrix(polyMatrix(0, 2, 2))
```

---

is.proper *Proper polynomial matrices*

Description

Tests the proper property of polynomial matrix. A polynomial matrix is proper if the associeted matrix has a full rank.

Usage

```
is.proper(pm)

is.column.proper(pm)

is.row.proper(pm)
```

Arguments

pm	a polyMatrix objects

Details

Polynomial matrix is column (row, full) proper (or reduced) if associated matrix has same rank as the number of column (row)

Value

True if object pm is a (row-/column-) proper matrix

Functions

- is.column.proper: tests if its argument is a column-proper matrix
- is.row.proper: tests if its argument is a row-proper matrix

Examples

```
pm <- parse.polyMatrix(
  "-1 + 7x     , x",
  " 3 - x + x^2, -1 + x^2 - 3 x^3"
)
is.column.proper(pm)
is.row.proper(pm)
is.proper(pm)
```

---

is.zero *Test if something is zero*

Description

Generic function to check if we can treat on object as being zero. For matrices the result is a matrix of the same size.

Usage

```
is.zero(x, eps = ZERO_EPS)

## S4 method for signature 'polynomial'
is.zero(x, eps = ZERO_EPS)

## S4 method for signature 'polyMatrix'
is.zero(x, eps = ZERO_EPS)
```

Arguments

x	The checked object
eps	Minimal numerical value which will not treat as zero

Details

Different type of objects can be treated as a zero in different ways:

- Numerical types can be compare by absolute value with eps.
- Customer types should define an an customer method.

By befault eps:

```
ZERO_EPS

## [1] 1e-05
```

Value

TRUE if the object can be treat as zero

Methods (by class)

- polynomial: a polynomail can be treated as zero if all its coefficients can be treated as zero
- polyMatrix: for a polunomial matrix every item is checked as polynomial

See Also

zero.round()

Examples

```
# numericals and matrices
is.zero(0) ## TRUE
is.zero(0.0001, eps=0.01) ## TRUE
is.zero(c(0, 1, 0)) ## TRUE, FALSE, TRUE
is.zero(matrix(c(1, 9, 0, 0), 2, 2))
# FALSE TRUE
# FALSE TRUE
```

```
# polynomials
is.zero(parse.polynomial("0.1 - 0.5 x")) ## FALSE
is.zero(parse.polynomial("0.0001 - 0.0005 x + 0.00002 x^2"), eps=0.01) ## TRUE
```

LCM *LCM for polynomial matrices*

Description

The least common multiple of polynomials or polynomial matrices.

Usage

```
LCM(...)

## S4 method for signature 'polyMatrix'
LCM(...)
```

Arguments

...	an list of polynomial objects

Methods (by class)

- polyMatrix: the least common multiple of polynomial matrices

See Also

polynomial implementation polynom::GCD() and GCD()

Examples

```
# LCM of polynomial matrix
LCM(parse.polyMatrix(
 "  1 - x, 1 - x^2, 1 + 2*x + x^2",
 "x - x^2,   1 + x, 1 - 2*x + x^2"
))  ## 0.25*x - 0.5*x^3 + 0.25*x^5
```

`matrix.degree` *Degree of each item of matrix*

Description

Returns a matrix obtained by applying a function `degree()` for each element of the matrix.

Usage

```
matrix.degree(x)

## S4 method for signature 'matrix'
matrix.degree(x)

## S4 method for signature 'polynomial'
matrix.degree(x)

## S4 method for signature 'polyMatrix'
matrix.degree(x)
```

Arguments

x	an R object

Details

Degree of each item is calculated using `degree()` which is defined for polynomials as the highest degree of the terms with non-zero coefficient.

For convenience this function is defined for any object, but returns zero for non polynomial objects.

Value

If the argument is a matrix, the result is a matrix of the same size containing the degrees of the matrix items.

For a numerical matrix the value is always a zero matrix of the same size

For a polynomial the value is the degree of the polynomial

Methods (by class)

- `matrix`: the degree of a numerical matrix is a zero matrix for compatibility
- `polynomial`: the degree of a polynomial
- `polyMatrix`: a matrix of degrees for each polynomial item of the source matrix

Examples

```
# numerical matrices
matrix.degree(matrix(1:6, 2, 3))
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0

# polynomials
matrix.degree(parse.polynomial("x + 1")) ## 1
matrix.degree(parse.polynomial("x^3 + 1")) ## 3
matrix.degree(parse.polynomial("1")) ## 0

# polynomial matrices
matrix.degree(parse.polyMatrix(
   "x; x^2 + 1",
   "0; 2x"))
##      [,1] [,2]
## [1,]    1    2
## [2,]    0    1
```

`minor` *Minor of matrix item*

Description

A minor of a matrix A is the determinant of some smaller square matrix, cut down from A by removing one or more of its rows and columns. Minors obtained by removing just one row and one column from square matrices (first minors).

Usage

```
minor(x, r, c)
```

Arguments

`x`	a matrix
`r, c`	row and column

`newton` *Build matrix of polynimal decomposition using Newton interpolation in Newton bais: (x-x_0), (x - x_0) * (x x_1)*

Description

Build matrix of polynimal decomposition using Newton interpolation in Newton bais: (x-x_0), (x - x_0) * (x x_1)

Usage

```
newton(C, points)
```

Arguments

C	Matrix of values of polinomials in columns
points	point in which the values of polynomials were got

Value

Matrix of coefficients in columns (from higher degree to lower)

---

parse.polyMatrix *Parse polynomial matrix from strings*

Description

This is a convenient way to input a polynomial matrix.

Usage

```
parse.polyMatrix(..., var = "x")
```

Arguments

...	string or strings to parse
var	variable character. Only lower latin characters are allowed except 'e' which is reseved for numbers

Details

Space and tabulation characters are ignored.

Row should be divided by new line "\n" or backslash "\" (TeX style).

Elements in each row can be divided by ",", ";" or "&" (TeX style)

For convenience, this function can accept multiple string. In this case each string will be treated as a new row.

This function accepts TeX matrix format.

Value

new polynomial matrix of polyMatrix class

Examples

```
parse.polyMatrix("       1, 2 + x",
                 "2 + 2x^2,    x^3")

# The function can suggest mistake position in case of invalid format
## Not run:
parse.polyMatrix(
    "1 + y &    2\\
        -2 &  x^2"
)
## Fail to parse polyMatrix: invalid term at position 2 in item [1, 1]

## End(Not run)
```

`parse.polynomial` *Parse polynomial from string*

Description

Parse string representation of polynomial into a polynomial object.

Usage

```
parse.polynomial(s, var = "x")
```

Arguments

`s`	an string for parsing
`var`	an variable name

Value

new polynomial as `polynom::polynomial` object

`polyMatrix` *Create polyMatrix object*

Description

This function will create polynomial object fromm coefficient matrix or signle value

Usage

```
polyMatrix(data, nrow, ncol, degree)
```

Arguments

data	an matrix in case of creation from coefficient matrices or an numer/polynomial
nrow	A numer of rows of matrix. If data is a matrix, default value is the number of rows of data matrix. In other case, it's a required parameter
ncol	Must be positibe. If data is a matrix, default value is the number of columns of data matrix. In other ccase, it's a required parameter.
degree	Degree of polynomials in coefficient matrix. Can't be negative. If data is polynomail, degree can be evaluated automatcal. In other case, default value is 0.

Details

A coefficient matrix is a matrix which contains matrices of coefficients from lower degree to higher side-by-side

Value

new polynomial matrix of polyMatrix class

---

polyMatrix-class *A class to represent matrix of polinomials*

Description

A class to represent matrix of polinomials

Usage

```
## S4 method for signature 'polyMatrix,numeric'
x[[i]]

## S4 method for signature 'polyMatrix'
det(x)

## S4 method for signature 'polyMatrix'
nrow(x)

## S4 method for signature 'polynomial'
nrow(x)

## S4 method for signature 'polyMatrix'
ncol(x)

## S4 method for signature 'polynomial'
ncol(x)

## S4 method for signature 'polyMatrix'
```

```
dim(x)

## S4 method for signature 'polyMatrix'
predict(object, newdata)

## S4 method for signature 'polyMatrix'
round(x, digits = 0)

## S4 method for signature 'polyMatrix'
show(object)

## S4 method for signature 'polyMatrix,missing,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'polyMatrix,missing,ANY,missing'
x[i, j]

## S4 method for signature 'polyMatrix,ANY,missing,missing'
x[i, j]

## S4 method for signature 'polyMatrix,logical,logical,missing'
x[i, j]

## S4 method for signature 'polyMatrix,logical,numeric,missing'
x[i, j]

## S4 method for signature 'polyMatrix,numeric,logical,missing'
x[i, j]

## S4 method for signature 'polyMatrix,numeric,numeric,missing'
x[i, j]

## S4 replacement method for signature 'polyMatrix,missing,missing,ANY'
x[i, j] <- value

## S4 replacement method for signature 'polyMatrix,missing,ANY,ANY'
x[i, j] <- value

## S4 replacement method for signature 'polyMatrix,ANY,missing,ANY'
x[i, j] <- value

## S4 replacement method for signature 'polyMatrix,numeric,numeric,numeric'
x[i, j] <- value

## S4 replacement method for signature 'polyMatrix,numeric,numeric,matrix'
x[i, j] <- value

## S4 replacement method for signature 'polyMatrix,numeric,numeric,polynomial'
```

```
x[i, j] <- value

## S4 replacement method for signature 'polyMatrix,numeric,numeric,polyMatrix'
x[i, j] <- value

## S4 method for signature 'polyMatrix,missing'
e1 + e2

## S4 method for signature 'polyMatrix,polyMatrix'
e1 + e2

## S4 method for signature 'polyMatrix,polynomial'
e1 + e2

## S4 method for signature 'polyMatrix,numeric'
e1 + e2

## S4 method for signature 'polyMatrix,matrix'
e1 + e2

## S4 method for signature 'ANY,polyMatrix'
e1 + e2

## S4 method for signature 'polyMatrix,polyMatrix'
e1 == e2

## S4 method for signature 'polyMatrix,polynomial'
e1 == e2

## S4 method for signature 'polyMatrix,matrix'
e1 == e2

## S4 method for signature 'polyMatrix,numeric'
e1 == e2

## S4 method for signature 'ANY,polyMatrix'
e1 == e2

## S4 method for signature 'polyMatrix,ANY'
e1 != e2

## S4 method for signature 'ANY,polyMatrix'
e1 != e2

## S4 method for signature 'polyMatrix,polyMatrix'
x %*% y

## S4 method for signature 'polyMatrix,matrix'
```

```
x %*% y

## S4 method for signature 'matrix,polyMatrix'
x %*% y

## S4 method for signature 'polyMatrix,numeric'
e1 * e2

## S4 method for signature 'polyMatrix,polynomial'
e1 * e2

## S4 method for signature 'polyMatrix,polyMatrix'
e1 * e2

## S4 method for signature 'ANY,polyMatrix'
e1 * e2

## S4 method for signature 'polyMatrix,polyMatrix'
e1 - e2

## S4 method for signature 'polyMatrix,ANY'
e1 - e2

## S4 method for signature 'ANY,polyMatrix'
e1 - e2
```

Arguments

x	an matrix object
i	the degree to extract matrix of coefficient
object	an R object
newdata	the value to evaluate
digits	integer indicating the number of decimal places (round) or significant digits (signif) to be used
j	column indeces
...	unused
drop	unused
value	new value
e1	an left operand
e2	an right operand
y	second argument

Methods (by generic)

- [[: get coefficient matrix by degree

- det: determinant of a polynomial matrix
- nrow: number of rows of a polynomial matrix
- nrow: an polynomial has only one row
- ncol: number of column of a polynomial matrix
- ncol: an polynomial has only one column
- dim: dimension of a polynomial matrix
- predict: value of polynomial matrix in point
- round: round of polynomial matrix is rounding of polynomial coefficients
- show: prints out a text representation of a polynomial matrix
- [: get matrix content
- [: get columns
- [: get rows
- [: get by logical index
- [: get by logical index and numerical indeces
- [: get by logical index and numerical indeces
- [: get by row and column indeces
- [<-: replace matrix content
- [<-: assign rows
- [<-: assign columns
- [<-: assign part of matrix with number
- [<-: assign part of matrix with another matrix
- [<-: assign part of matrix with polynomial
- [<-: assign part of matrix with another polynomial matrix
- +: summation with polynomial matrix
- +: summation of polynomial matrices
- +: summation of polynomial matrix and scalar polynomial
- +: summation of polynomial matrix and scalar nummber
- +: summation of polynomial matrix and numerical matrix
- +: summation of polynomial matrix
- ==: equal operator for two polinomial matrices, result is a boolean matrix
- ==: equal operator for polinomail matrix and polinomail, result is a matrix
- ==: equal operator for polinomial and numerical matrices
- ==: equal operator for polinomial matrix and number, result is a matrix
- ==: equal operator for aby object and polinomial matrix
- !=: not equal operator
- !=: not equal operator
- %*%: matrix multiplicatoin of polynomial matrices

- %*%: matrix multiplicatoin of polynomial and numerical matrices
- %*%: matrix multiplicatoin of numerical and polynomial matrices
- *: scalar multiplication with number
- *: scalar multiplication with polynomial
- *: scalar multiplication of polynomial mattrices elementwise
- *: scalar multiplication
- -: substractioin
- -: substractioin
- -: substractioin

Slots

`coef` A matrix of coefficients which are joined into one matrix from lower degree to higher

`ncol` Actual number of columns in the polynomial matrix

Examples

```
# create a new polynomial matrix by parsing strings
pm <- parse.polyMatrix(
     "x; 1 + x^2; 3 x - x^2",
     "1; 1 + x^3; - x + x^3"
)

# get coefficient matrix for degree 0
pm[[0]]
##      [,1] [,2] [,3]
## [1,]    0    1    0
## [2 ]    1    1    0
# get coefficient matrix for degree 1
pm[[1]]
##      [,1] [,2] [,3]
## [1,]    1    0    3
## [2 ]    0    0   -1


# dimensions
nrow(pm) ## 2


ncol(pm) ## 3


dim(pm) ## [1] 2 3


# round
round(parse.polyMatrix(
  "      1.0001 - x,            1 - x^2, 1 + 2.0003*x + x^2",
  "0.0001 + x - x^2, 1 + x + 0.0001 x^2, 1 - 2*x + x^2"
```

```
))
##           [,1]      [,2]          [,3]
## [1,]     1 - x   1 - x^2   1 + 2x + x^2
## [2,]   x - x^2     1 + x   1 - 2x + x^2


# print out a polynomial matrix
show(parse.polyMatrix(
  "      1.0001 - x,          1 - x^2, 1 + 2.0003*x + x^2",
  "0.0001 + x - x^2,            1 + x, 1 - 2*x + x^2",
  "        12.3 x^3,  2 + 3.5 x + x^4, -0.7 + 1.6e-3 x^3"
))
##                   [,1]             [,2]                [,3]
## [1,]        1.0001 - x          1 - x^2   1 + 2.0003x + x^2
## [2,]   1e-04 + x - x^2            1 + x        1 - 2x + x^2
## [3,]           12.3x^3   2 + 3.5x + x^4    -0.7 + 0.0016x^3
```

---

polyMatrix.apply *Apply for polynomial matrix*

Description

Apply function to each element of matrix

Usage

```
polyMatrix.apply(x, f)
```

Arguments

x	an polynomial matrix
f	an function with only one argument

---

t,polyMatrix-method *Polynomial matrix transpose*

Description

Given a polyMatrix, t returns the transpose of x

Usage

```
## S4 method for signature 'polyMatrix'
t(x)
```

Arguments

x	a polyMatrix

See Also

`base::t()` for numerical matrix tranpose

Examples

```
pm <- parse.polyMatrix("1, x, x^2",
                       "x, 1, x^3")
t(pm)
##        [,1]   [,2]
## [1,]      1      x
## [2,]      x      1
## [3,]    x^2    x^3
```

---

tr *Trace of a 'matrix' or 'polyMatrix' class matrix*

Description

Trace of a matrix is the sum of the diagonal elements of the given matrix.

Usage

```
tr(x)
```

Arguments

x	an matrix or a polynomial matrix

Details

If the given matrix is a polynomial matrix, the result will be a polynomial.

Value

Returns the trace of the given matrix as a number or a polynomial.

Examples

```
# numerical matrices
m <- matrix(1:12, 3, 4)
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
tr(m)  ## 15
```

```
# polynomial matrix
pm <- parse.polyMatrix(
  "-3 + x^2, 2 + 4 x,  -x^2",
  "       1,       2, 3 + x",
  "     2*x,       0, 2 - 3 x"
)
tr(pm)  ## 1 - 3*x + x^2
```

`triang_Interpolation` *Triangularization of a polynomial matrix by interpolation method*

Description

The parameters `point_vector`, `round_digits` can significantly affect the result.

Usage

```
triang_Interpolation(
  pm,
  point_vector,
  round_digits = 5,
  eps = .Machine$double.eps^0.5
)
```

Arguments

`pm`	source polynimial matrix
`point_vector`	vector of interpolation points
`round_digits`	we will try to round result on each step
`eps`	calculation zero errors

Details

Default value of ‘eps“ usually is enought to determintate real zeros.

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 offers a solution of the triangulrization by the Interpolation method, described in the article of Labhalla-Lombardi-Marlin (1996).

Value

Tranfortmaiton matrix

`triang_Sylvester` *Triangularization of a polynomial matrix by Sylvester method*

Description

The function `triang_Sylvester` triangularize the given polynomial matrix.

Usage

```
triang_Sylvester(pm, u, eps = ZERO_EPS)
```

Arguments

`pm`	an polynomial matrix to triangularize
`u`	the minimal degree of the triangularizator multiplicator
`eps`	threshold of non zero coefficients

Details

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.

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 triangulrization 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

References

Salah Labhalla, Henri Lombardi, Roger Marlin: Algorithm de calcule de la reduction de Hermite d'une matrice a coefficients polynomiaux, Theoretical Computer Science 161 (1996) pp 69-92

`zero.round` *Rounds object to zero if it's too small*

Description

Rounds object to zero if it's too small

Usage

```
zero.round(x, eps = ZERO_EPS)

## S4 method for signature 'polynomial'
zero.round(x, eps = ZERO_EPS)

## S4 method for signature 'polyMatrix'
zero.round(x, eps = ZERO_EPS)
```

Arguments

`x`	an R object
`eps`	Minimal numerical value which will not treat as zero

Details

By default eps:

```
ZERO_EPS

## [1] 1e-05
```

Methods (by class)

- `polynomial`: rounding of a polynomial means rounding of each coefficient
- `polyMatrix`: rounding of a polynomial matrix

See Also

`is.zero()`

Examples

```
# numerical
zero.round(1)  ## 1
zero.round(0)  ## 0
zero.round(0.1, eps=0.5) ## 0
zero.round(c(1, 0, .01, 1e-10)) ##  1.00 0.00 0.01 0.00
```

```
# polynomials
zero.round(parse.polynomial("0.1 + x + 1e-7 x^2")) ## 0.1 + x
zero.round(parse.polynomial("0.1 + x + 1e-7 x^2"), eps=0.5) ## x


# polynomial matrix
zero.round(parse.polyMatrix(
  "1 + 0.1 x, 10 + x + 3e-8 x^2, 1e-8",
  "0.1 + x^2,     .1 + 1e-8 x^4, 1e-8 x^5"
))
##             [,1]     [,2]   [,3]
## [1,]    1 + 0.1x   10 + x      0
## [2,]   0.1 + x^2      0.1      0

zero.round(parse.polyMatrix(
  "1 + 0.1 x, 10 + x + 3e-8 x^2, 1e-8",
  "0.1 + x^2,     .1 + 1e-8 x^4, 1e-8 x^5"
), eps=0.5)
##        [,1]     [,2]   [,3]
## [1,]      1   10 + x      0
## [2,]    x^2        0      0
```

`zero_lead_hyp_rows` *Get zero lead hyper rows of size sub_nrow of matrix M*

Description

Get zero lead hyper rows of size sub_nrow of matrix M

Usage

```
zero_lead_hyp_rows(M, sub_nrow, esp = ZERO_EPS)
```

Arguments

`M`	Numerical matrix
`sub_nrow`	Size of hyper row
`esp`	Machine epsilon to determinate zeros

Value

vector of idx of hyperrows, NaN for columns without zeros

`zero_lead_rows` *Get zero lead rows of matrix M*

Description

Get zero lead rows of matrix M

Usage

```
zero_lead_rows(M, eps = ZERO_EPS)
```

Arguments

`M`	Numerical matrix
`eps`	Machine epsilon to determinate zeros

Value

vector of idx (length is equal to columm number), NULL in case of error

Index