

# Package ‘EigenR’

May 18, 2022

**Type** Package

**Title** Complex Matrix Algebra with 'Eigen'

**Version** 1.2.3

**Author** Stéphane Laurent

**Maintainer** Stéphane Laurent <laurent\_step@outlook.fr>

**Description** Matrix algebra using the 'Eigen' C++ library: determinant, rank, inverse, pseudo-inverse, kernel and image, QR decomposition, Cholesky decomposition, linear least-squares problems. Also provides matrix functions such as exponential, logarithm, power, sine and cosine. Complex matrices are supported.

**License** GPL-3

**Imports** Rcpp (>= 1.0.5)

**LinkingTo** Rcpp, RcppEigen

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**URL** <https://github.com/stla/EigenR>

**BugReports** <https://github.com/stla/EigenR/issues>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-05-18 09:00:02 UTC

## R topics documented:

Eigen_absdet . . . . .	2
Eigen_chol . . . . .	3
Eigen_cos . . . . .	3
Eigen_cosh . . . . .	4
Eigen_det . . . . .	5
Eigen_exp . . . . .	5
Eigen_inverse . . . . .	6
Eigen_isInjective . . . . .	6

Eigen_isInvertible . . . . .	7
Eigen_isSurjective . . . . .	7
Eigen_kernel . . . . .	8
Eigen_kernelDimension . . . . .	8
Eigen_log . . . . .	9
Eigen_logabsdet . . . . .	10
Eigen_lsSolve . . . . .	10
Eigen_pinverse . . . . .	11
Eigen_pow . . . . .	12
Eigen_QR . . . . .	12
Eigen_range . . . . .	13
Eigen_rank . . . . .	13
Eigen_sin . . . . .	14
Eigen_sinh . . . . .	14
Eigen_sqrt . . . . .	15
Eigen_UtDU . . . . .	15
SparseMatrix . . . . .	16

## Index 18

---

Eigen_absdet	<i>Absolute value of the determinant</i>
--------------	--

---

### Description

Absolute value of the determinant of a real matrix.

### Usage

```
Eigen_absdet(M)
```

### Arguments

M                    a *real* square matrix

### Value

The absolute value of the determinant of M.

### Note

‘Eigen\_absdet(M)’ is not faster than ‘abs(Eigen\_det(M))’.

### Examples

```
set.seed(666L)
M <- matrix(rpois(25L, 1), 5L, 5L)
Eigen_absdet(M)
```

---

Eigen_chol	<i>Cholesky decomposition of a matrix</i>
------------	---

---

**Description**

Cholesky decomposition of a symmetric or Hermitian matrix.

**Usage**

```
Eigen_chol(M)
```

**Arguments**

M a square symmetric/Hermitian positive-definite matrix or [SparseMatrix](#), real/complex

**Details**

Symmetry is not checked; only the lower triangular part of M is used.

**Value**

The upper triangular factor of the Cholesky decomposition of M.

**Examples**

```
M <- rbind(c(5,1), c(1,3))
U <- Eigen_chol(M)
t(U) %*% U # this is `M`
# a Hermitian example:
A <- rbind(c(1,1i), c(1i,2))
( M <- A %*% t(Conj(A)) )
try(chol(M)) # fails
U <- Eigen_chol(M)
t(Conj(U)) %*% U # this is `M`
# a sparse example
M <- asSparseMatrix(diag(1:5))
Eigen_chol(M)
```

---

Eigen_cos	<i>Matrix cosine</i>
-----------	----------------------

---

**Description**

Matrix cosine of a real or complex square matrix.

**Usage**

```
Eigen_cos(M)
```

**Arguments**

M a square matrix, real or complex

**Value**

The matrix cosine of M.

**Examples**

```
library(EigenR)
M <- toeplitz(c(1,2,3))
cosM <- Eigen_cos(M)
sinM <- Eigen_sin(M)
cosM %% cosM + sinM %% sinM # identity matrix
```

---

Eigen\_cosh

*Matrix hyperbolic cosine*

---

**Description**

Matrix hyperbolic cosine of a real or complex square matrix.

**Usage**

```
Eigen_cosh(M)
```

**Arguments**

M a square matrix, real or complex

**Value**

The matrix hyperbolic cosine of M.

**Examples**

```
library(EigenR)
M <- toeplitz(c(1,2,3))
Eigen_cosh(M)
(Eigen_exp(M) + Eigen_exp(-M)) / 2 # identical
```

---

Eigen_det	<i>Determinant of a matrix</i>
-----------	--------------------------------

---

**Description**

Determinant of a real or complex matrix.

**Usage**

```
Eigen_det(M)
```

**Arguments**

M a square matrix or [SparseMatrix](#), real or complex

**Value**

The determinant of M.

**Examples**

```
set.seed(666)
M <- matrix(rpois(25, 1), 5L, 5L)
Eigen_det(M)
# determinants of complex matrices are supported:
Eigen_det(M + 1i * M)
# as well as determinants of sparse matrices:
Eigen_det(asSparseMatrix(M))
Eigen_det(asSparseMatrix(M + 1i * M))
```

---

Eigen_exp	<i>Exponential of a matrix</i>
-----------	--------------------------------

---

**Description**

Exponential of a real or complex square matrix.

**Usage**

```
Eigen_exp(M)
```

**Arguments**

M a square matrix, real or complex

**Value**

The exponential of M.

---

Eigen_inverse	<i>Inverse of a matrix</i>
---------------	----------------------------

---

**Description**

Inverse of a real or complex matrix.

**Usage**

```
Eigen_inverse(M)
```

**Arguments**

M                    an invertible square matrix, real or complex

**Value**

The inverse matrix of M.

---

Eigen_isInjective	<i>Check injectivity</i>
-------------------	--------------------------

---

**Description**

Checks whether a matrix represents an injective linear map (i.e. has trivial kernel).

**Usage**

```
Eigen_isInjective(M)
```

**Arguments**

M                    a matrix, real or complex

**Value**

A Boolean value indicating whether M represents an injective linear map.

**Examples**

```
set.seed(666L)
M <- matrix(rpois(35L, 1), 5L, 7L)
Eigen_isInjective(M)
```

---

Eigen\_isInvertible      *Check invertibility*

---

**Description**

Checks whether a matrix is invertible.

**Usage**

```
Eigen_isInvertible(M)
```

**Arguments**

M                      a matrix, real or complex

**Value**

A Boolean value indicating whether M is invertible.

**Examples**

```
set.seed(666L)
M <- matrix(rpois(25L, 1), 5L, 5L)
Eigen_isInvertible(M)
```

---

Eigen\_isSurjective      *Check surjectivity*

---

**Description**

Checks whether a matrix represents a surjective linear map.

**Usage**

```
Eigen_isSurjective(M)
```

**Arguments**

M                      a matrix, real or complex

**Value**

A Boolean value indicating whether M represents a surjective linear map.

**Examples**

```
set.seed(666L)
M <- matrix(rpois(35L, 1), 7L, 5L)
Eigen_isSurjective(M)
```

---

Eigen_kernel	<i>Kernel of a matrix</i>
--------------	---------------------------

---

**Description**

Kernel (null-space) of a real or complex matrix.

**Usage**

```
Eigen_kernel(M, method = "COD")
```

**Arguments**

M	a matrix, real or complex
method	one of "COD" or "LU"; the faster method depends on the size of the matrix

**Value**

A basis of the kernel of M. With method = "COD", the basis is orthonormal, while it is not with method = "LU".

**Examples**

```
set.seed(666)
M <- matrix(rgamma(30L, 12, 1), 10L, 3L)
M <- cbind(M, M[,1]+M[,2], M[,2]+2*M[,3])
# basis of the kernel of `M`:
Eigen_kernel(M, method = "LU")
# orthonormal basis of the kernel of `M`:
Eigen_kernel(M, method = "COD")
```

---

Eigen_kernelDimension	<i>Dimension of kernel</i>
-----------------------	----------------------------

---

**Description**

Dimension of the kernel of a matrix.

**Usage**

```
Eigen_kernelDimension(M)
```

**Arguments**

M	a matrix, real or complex
---	---------------------------

**Value**

An integer, the dimension of the kernel of M.

**See Also**

[Eigen\\_isInjective](#), [Eigen\\_kernel](#).

**Examples**

```
set.seed(666L)
M <- matrix(rpois(35L, 1), 5L, 7L)
Eigen_kernelDimension(M)
```

---

Eigen_log	<i>Logarithm of a matrix</i>
-----------	------------------------------

---

**Description**

Logarithm of a real or complex square matrix, when possible.

**Usage**

```
Eigen_log(M)
```

**Arguments**

M                    a square matrix, real or complex

**Details**

The logarithm of a matrix does not always exist. See [matrix logarithm](#).

**Value**

The logarithm of M.

---

Eigen_logabsdet	<i>Logarithm of the absolute value of the determinant</i>
-----------------	---

---

**Description**

Logarithm of the absolute value of the determinant of a real matrix.

**Usage**

```
Eigen_logabsdet(M)
```

**Arguments**

M                    a *real* square matrix

**Value**

The logarithm of the absolute value of the determinant of M.

**Note**

'Eigen\_logabsdet(M)' is not faster than 'log(abs(Eigen\_det(M)))'.

**Examples**

```
set.seed(666L)
M <- matrix(rpois(25L, 1), 5L, 5L)
Eigen_logabsdet(M)
```

---

Eigen_lsSolve	<i>Linear least-squares problems</i>
---------------	--------------------------------------

---

**Description**

Solves a linear least-squares problem.

**Usage**

```
Eigen_lsSolve(A, b, method = "cod")
```

**Arguments**

A                    a  $n \times p$  matrix, real or complex  
b                    a vector of length  $n$  or a matrix with  $n$  rows, real or complex  
method              the method used to solve the problem, either "svd" (based on the SVD decomposition) or "cod" (based on the complete orthogonal decomposition)

**Value**

The solution  $X$  of the least-squares problem  $AX \approx b$  (similar to `lm.fit(A, b)$coefficients`). This is a matrix if  $b$  is a matrix, or a vector if  $b$  is a vector.

**Examples**

```
set.seed(129)
n <- 7; p <- 2
A <- matrix(rnorm(n * p), n, p)
b <- rnorm(n)
lsfit <- Eigen_lsSolve(A, b)
b - A %>% lsfit # residuals
```

Eigen\_pinverse

*Pseudo-inverse of a matrix***Description**

Pseudo-inverse of a real or complex matrix (Moore-Penrose generalized inverse).

**Usage**

```
Eigen_pinverse(M)
```

**Arguments**

`M` a matrix, real or complex, not necessarily square

**Value**

The pseudo-inverse matrix of  $M$ .

**Examples**

```
library(EigenR)
M <- rbind(
  toeplitz(c(3, 2, 1)),
  toeplitz(c(4, 5, 6))
)
Mplus <- Eigen_pinverse(M)
all.equal(M, M %>% Mplus %>% M)
all.equal(Mplus, Mplus %>% M %>% Mplus)
#' a complex matrix
A <- M + 1i * M[, c(3L, 2L, 1L)]
Aplus <- Eigen_pinverse(A)
Aplus <- A %>% Aplus
all.equal(Aplus, t(Conj(Aplus))) #' `A %>% Aplus` is Hermitian
AplusA <- Aplus %>% A
all.equal(AplusA, t(Conj(AplusA))) #' `Aplus %>% A` is Hermitian
```

---

Eigen_pow	<i>Matricial power</i>
-----------	------------------------

---

**Description**

Matricial power of a real or complex square matrix, when possible.

**Usage**

```
Eigen_pow(M, p)
```

**Arguments**

M	a square matrix, real or complex
p	a number, real or complex, the power exponent

**Details**

The power is defined with the help of the exponential and the logarithm. See [matrix power](#).

**Value**

The matrix M raised at the power p.

---

Eigen_QR	<i>QR decomposition of a matrix</i>
----------	-------------------------------------

---

**Description**

QR decomposition of a real or complex matrix.

**Usage**

```
Eigen_QR(M)
```

**Arguments**

M	a matrix, real or complex
---	---------------------------

**Value**

A list with the Q matrix and the R matrix.

**Examples**

```
M <- cbind(c(1,2,3), c(4,5,6))
x <- Eigen_QR(M)
x$Q %*% x$R
```

---

Eigen_range	<i>Range of a matrix</i>
-------------	--------------------------

---

**Description**

Range (column-space, image, span) of a real or complex matrix.

**Usage**

```
Eigen_range(M, method = "QR")
```

**Arguments**

M	a matrix, real or complex
method	one of "LU", "QR", or "COD"; the "LU" method is faster

**Value**

A basis of the range of M. With method = "LU", the basis is not orthonormal, while it is with method = "QR" and method = "COD".

---

Eigen_rank	<i>Rank of a matrix</i>
------------	-------------------------

---

**Description**

Rank of a real or complex matrix.

**Usage**

```
Eigen_rank(M)
```

**Arguments**

M	a matrix, real or complex
---	---------------------------

**Value**

The rank of M.

---

**Eigen\_sin***Matrix sine*

---

**Description**

Matrix sine of a real or complex square matrix.

**Usage**

Eigen\_sin(M)

**Arguments**

M                    a square matrix, real or complex

**Value**

The matrix sine of M.

---

**Eigen\_sinh***Matrix hyperbolic sine*

---

**Description**

Matrix hyperbolic sine of a real or complex square matrix.

**Usage**

Eigen\_sinh(M)

**Arguments**

M                    a square matrix, real or complex

**Value**

The matrix hyperbolic sine of M.

**Examples**

```
library(EigenR)
M <- toeplitz(c(1,2,3))
Eigen_sinh(M)
(Eigen_exp(M) - Eigen_exp(-M)) / 2 # identical
```

---

Eigen_sqrt	<i>Square root of a matrix</i>
------------	--------------------------------

---

**Description**

Square root of a real or complex square matrix, when possible.

**Usage**

```
Eigen_sqrt(M)
```

**Arguments**

M a square matrix, real or complex

**Details**

See [matrix square root](#).

**Value**

A square root of M.

**Examples**

```
# Rotation matrix over 60 degrees:  
M <- cbind(c(cos(pi/3), sin(pi/3)), c(-sin(pi/3), cos(pi/3)))  
# Its square root, the rotation matrix over 30 degrees:  
Eigen_sqrt(M)
```

---

Eigen_UtDU	<i>'UtDU' decomposition of a matrix</i>
------------	---

---

**Description**

Cholesky-'UtDU' decomposition of a symmetric or Hermitian matrix.

**Usage**

```
Eigen_UtDU(M)
```

**Arguments**

M a square symmetric/Hermitian positive or negative semidefinite matrix, real/complex

**Details**

Symmetry is not checked; only the lower triangular part of M is used.

**Value**

The Cholesky-'UtDU' decomposition of M in a list (see example).

**Examples**

```
x <- matrix(c(1:5, (1:5)^2), 5, 2)
x <- cbind(x, x[, 1] + 3*x[, 2])
M <- crossprod(x)
UtDU <- Eigen_UtDU(M)
U <- UtDU$U
D <- UtDU$D
perm <- UtDU$perm
UP <- U[, perm]
t(UP) %*% diag(D) %*% UP # this is `M`
```

---

 SparseMatrix

*Sparse matrix*


---

**Description**

Constructs a sparse matrix, real or complex.

**Usage**

```
SparseMatrix(i, j, Mij, nrows, ncols)
```

```
## S3 method for class 'SparseMatrix'
print(x, ...)
```

```
asSparseMatrix(M)
```

**Arguments**

<code>i, j</code>	indices of the non-zero coefficients
<code>Mij</code>	values of the non-zero coefficients; must be a vector of the same length as <code>i</code> and <code>j</code> or a single number which will be recycled
<code>nrows, ncols</code>	dimensions of the matrix
<code>x</code>	a SparseMatrix object
<code>...</code>	ignored
<code>M</code>	a matrix, real or complex

**Value**

A list with the class `SparseMatrix`.

**Examples**

```
set.seed(666)
( M <- matrix(rpois(50L, 1), 10L, 5L) )
asSparseMatrix(M)
```

# Index

`asSparseMatrix (SparseMatrix)`, 16

`Eigen_absdet`, 2

`Eigen_chol`, 3

`Eigen_cos`, 3

`Eigen_cosh`, 4

`Eigen_det`, 5

`Eigen_exp`, 5

`Eigen_inverse`, 6

`Eigen_isInjective`, 6, 9

`Eigen_isInvertible`, 7

`Eigen_isSurjective`, 7

`Eigen_kernel`, 8, 9

`Eigen_kernelDimension`, 8

`Eigen_log`, 9

`Eigen_logabsdet`, 10

`Eigen_lsSolve`, 10

`Eigen_pinverse`, 11

`Eigen_pow`, 12

`Eigen_QR`, 12

`Eigen_range`, 13

`Eigen_rank`, 13

`Eigen_sin`, 14

`Eigen_sinh`, 14

`Eigen_sqrt`, 15

`Eigen_UtDU`, 15

`print.SparseMatrix (SparseMatrix)`, 16

`SparseMatrix`, 3, 5, 16