

# Package ‘complexplus’

July 22, 2025

**Type** Package  
**Title** Functions of Complex or Real Variable  
**Version** 2.1  
**Date** 2017-05-18  
**Author** Albert Dorador and Uffe Høgsbro Thygesen  
**Maintainer** Albert Dorador <albert.dorador@estudiant.upc.edu>  
**Description** Extension of several functions to the complex domain, including the matrix exponential and logarithm, and the determinant.  
**Depends** R (>= 3.1.0)  
**Imports** expm (>= 0.999-2), Matrix (>= 1.2-6)  
**License** GPL-2  
**ByteCompile** true  
**Encoding** UTF-8  
**RoxygenNote** 6.0.1  
**NeedsCompilation** no  
**Repository** CRAN  
**Date/Publication** 2017-05-18 08:26:26 UTC

## Contents

|                      |          |
|----------------------|----------|
| Det . . . . .        | 2        |
| Imzap . . . . .      | 3        |
| matchclass . . . . . | 3        |
| matexp . . . . .     | 4        |
| matlog . . . . .     | 5        |
| <b>Index</b>         | <b>7</b> |

---

Det*Compute the Determinant of a Matrix*

---

## Description

Det computes the determinant of a square matrix. This function first checks whether the matrix is full rank or not; if not, the value 0 is returned. This avoids relatively frequent numerical errors that produce a non-zero determinant when in fact it is zero. Only if the matrix is full rank does the algorithm proceed to compute the determinant. If the matrix is complex, the determinant is computed as the product of the eigenvalues; if the matrix is real, Det calls the base function det for maximum efficiency.

## Usage

Det(M)

## Arguments

M                      a square matrix, real or complex.

## Value

The determinant of M.

## Author(s)

Albert Dorador

## Examples

```
A <- matrix(c(1, 2, 2+3i, 5), ncol = 2) #complex matrix
B <- matrix(1:4, ncol = 2) #real matrix
S <- matrix(c(3, 4+3i, 0, 0), ncol = 2) #Singular matrix

Det(A)
Det(B)
Det(S)
```

---

**Imzap***Rounding of Null Imaginary Part of a Complex Number*

---

**Description**

imaginary parts with values very close to 0 are 'zapped', i.e. treated as 0. Therefore, the number becomes real and changes its class from complex to numeric.

**Usage**

```
Imzap(x, tol = 1e-06)
```

**Arguments**

**x** a scalar or vector, real or complex.

**tol** a tolerance,  $10^{-6}$  by default. Prevents possible numerical problems. Can be set to 0 if desired.

**Author(s)**

Albert Dorador

**Examples**

```
x1 <- 1:100
x2 <- c(1:98,2+3i,0-5i)
x3 <- c(1:98,2+0i,7-0i)
x4 <- complex(real = rnorm(100), imaginary = rnorm(100))

Imzap(x1) # innocuous with real vectors
Imzap(x2) # 1 single element is enough to turn the vector into complex
Imzap(x3) # removes extra 0i and changes class from complex to numeric
Imzap(x4) # innocuous with complex vectors with non-null complex part
```

---

**matclass***Matrix Class*

---

**Description**

matclass gives the name of the class of the matrix. If the class of the matrix elements is numeric or integer, the matrix class is real by default.

**Usage**

```
matclass(M, real = TRUE)
```

**Arguments**

|                   |                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <code>M</code>    | a matrix, of any type and dimensions.                                                                         |
| <code>real</code> | a logical value. Set to FALSE if you want to opt out of the default treatment of numeric or integer elements. |

**Author(s)**

Albert Dorador

**Examples**

```
A <- matrix(c(1, 2, 2+3i, 5), ncol = 2) # complex matrix
B <- matrix(1:4, ncol = 2) # real matrix
C <- matrix(c(TRUE, TRUE, FALSE, FALSE), ncol = 2) # boolean matrix
D <- matrix(c('a', 'b', 'c', 'd'), ncol = 2) # character matrix
E <- matrix(c(1,2+3i,'a',TRUE),ncol=2) # 1 single character makes it a character matrix

matclass(A)
matclass(B)
matclass(B,FALSE)
matclass(C)
matclass(D)
matclass(E)
```

---

matexp

---

*Matrix Exponential*


---

**Description**

matexp computes the exponential of a square matrix  $A$  i.e.  $\exp(A)$ .

**Usage**

```
matexp(A, ...)
```

**Arguments**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>A</code>   | a square matrix, real or complex.          |
| <code>...</code> | arguments passed to or from other methods. |

**Details**

This function adapts function `expm` from package **expm** to be able to handle complex matrices, by decomposing the original matrix into real and purely imaginary matrices and creating a real block matrix that function `expm` can successfully process. If the original matrix is real, `matexp` calls `expm` directly for maximum efficiency.

**Value**

The matrix exponential of A. Method used may be chosen from the options available in expm.

**Author(s)**

Uffe Høgsbro Thygesen

**See Also**

[expm](#)

**Examples**

```
A <- matrix(c(1, 2, 2+3i, 5), ncol = 2) # complex matrix
B <- matrix(1:4, ncol = 2) # real matrix

matexp(A)
matexp(A, "Ward77") # uses Ward77's method in function expm
matexp(B)
matexp(B, "Taylor") # uses Taylor's method in function expm
```

---

matlog

---

*Matrix Logarithm*


---

**Description**

matlog computes the (principal) matrix logarithm of a square matrix.

**Usage**

```
matlog(A, ...)
```

**Arguments**

|     |                                            |
|-----|--------------------------------------------|
| A   | a square matrix, real or complex.          |
| ... | arguments passed to or from other methods. |

**Details**

This function adapts function logm from package **expm** to be able to handle complex matrices, by decomposing the original matrix into real and purely imaginary matrices and creating a real block matrix that function logm can successfully process. If the original matrix is real, matlog calls logm directly for maximum efficiency. Hence, for real matrices, matlog can compute the matrix logarithm in the same instances as logm; for complex matrices, matlog can compute the matrix logarithm as long as all real eigenvalues are positive: zero eigenvalues imply singularity (and therefore the log does not exist) and negative eigenvalues can be problematic as it may be hard and numerically unstable to calculate Jordan blocks. See references below.

**Value**

The matrix logarithm of A. Method used may be chosen from the options available in logm.

**Author(s)**

Albert Dorador

**References**

For more on the matrix logarithm, visit [https://en.wikipedia.org/wiki/Logarithm\\_of\\_a\\_matrix](https://en.wikipedia.org/wiki/Logarithm_of_a_matrix)

**See Also**

[logm](#)

**Examples**

```
A <- matrix(c(1, 2, 2+3i, 5), ncol = 2) # complex matrix
B <- matrix(c(2, 0, 3, 4), ncol = 2) # real matrix with positive eigenvalues

matlog(A)
matlog(A, "Eigen") # uses Eigen method in function logm
matlog(B)
matlog(B, "Eigen") # uses Eigen method in function logm
```

# Index

Det, [2](#)

expm, [5](#)

Imzap, [3](#)

logm, [6](#)

matchclass, [3](#)

matexp, [4](#)

matlog, [5](#)