

Package ‘MPCR’

November 26, 2023

Type Package

Title Multi Precision Computing

Version 1.0

Date 2023-11-12

Author David Helmy [aut, cph], Sameh Abdulah [cre, cph], KAUST King Abdullah University of Science and Technology [fnd, cph], Brightskies [cph]

Maintainer Sameh Abdulah <sameh.abdulah@kaust.edu.sa>

Description Provides new data-structure support for multi- and mixed-precision for R users. The package supports 16-bit, 32-bit, and 64-bit operations. To the best of our knowledge, 'MPCR' differs from the currently available packages in the following: 'MPCR' introduces a new data structure that supports three different precisions (16-bit, 32-bit, and 64-bit), allowing for optimized memory allocation based on the desired precision. This feature offers significant advantages in-memory optimization. 'MPCR' extends support to all basic linear algebra methods across different precisions. 'MPCR' maintains a consistent interface with normal R functions, allowing for seamless code integration and a user-friendly experience.

License GPL (>= 3)

Imports methods, Rcpp (>= 1.0.9)

Depends R (>= 3.6.0)

RoxygenNote 7.2.3

SystemRequirements CMake (>=3.20) , C++ (>= 11)

NeedsCompilation yes

OS_type unix

URL <https://github.com/stsds/MPCR>

R topics documented:

01-MPCR	2
02-Converters	4
03-Arithmetic	5
04-Comparison	6
05-Copy	7

06-Dimensions	8
07-Extract-Replace	9
08-Concatenate	10
09-Bind	10
10-Diagonal	11
11-Extremes	12
12-Log	12
13-Mathis	13
14-Miscmath	14
15-NA's	15
16-Replicate	16
17-Round	16
18-Scale	17
19-Sweep	18
20-Special Math	19
21-Trig	19
22-Hyperbolic	20
23-Transpose	21
24-Check precision	22
25-Metadata	22
26-Print	23
27-Cholesky decomposition	24
28-Cholesky inverse	25
29-Crossprod	26
30-Eigen decomposition	27
31-Symmetric	27
32-Norm	28
33-QR decomposition	29
34-Reciprocal condition	30
35-Solve	31
36-Singular value decomposition	31
37-Back/Forward solve	32
38-MPCR GEMM	33
39-MPCR TRSM	34

Index	36
--------------	-----------

01-MPCR

MPCR S4 Class

Description

MPCR is a multi-precision vector/matrix, that enables the creation of vector/matrix with three different precisions (16-bit (half), 32-bit(single), and 64-bit(double)).

Value

MPCR object (constructor - accessors - methods)

Constructor

new Creates a new instance of zero values of the MPCR class. `new(MPCR, size, "precision")`

size The total number of values for which memory needs to be allocated.

precision String to indicate the precision of MPCR object ("half", "single", or "double").

Accessors

The following accessors can be used to get the values of the slots:

IsMatrix Boolean to indicate whether the MPCR object is a vector or matrix.

Size Total number of elements inside the object, (row*col) in the case of matrix, and number of elements in the case of vector.

Row Number of rows.

Col Number of cols.

Methods

The following methods are available for objects of class MPCR:

PrintValues: `PrintValues()`: Prints all the values stored in the matrix or vector, along with metadata about the object.

ToMatrix: `ToMatrix(row, col)`: Changes the object representation to match the new dimensions, no memory overhead.

ToVector: `ToVector()`: Changes the MPCR matrix to vector, no memory overhead.

Examples

```
# Example usage of the class and its methods
library(MPCR)
MPCR_object <- new(MPCR, 50, "single")

MPCR_object$ToMatrix(5, 10)
MPCR_object$Row      #5
MPCR_object$Col      #10
MPCR_object$Size     #50
MPCR_object$IsMatrix #TRUE

MPCR_object$PrintValues()
MPCR_object$ToVector()

MPCR_object
```

Description

Converters from R to MPCR objects and vice-versa.

Value

An MPCR or R numeric vector/matrix.

MPCR Converter

Convert R object to MPCR object.

MPCR converters:

`as.MPCR(data, nrow = 0, ncol = 0, precision)`: Converts R object to MPCR object.

`data` R matrix/vector.

`nrow` Number of rows of the new MPCR matrix, **default = zero** which means a vector will be created.

`ncol` Number of cols of the new MPCR matrix, **default = zero** which means a vector will be created.

`precision` String indicates the precision of the new MPCR object (half, single, or double).

R Converter

Convert an MPCR object to R object.

R vector converter:

`MPCR.ToNumericVector(x)`: Converts an MPCR object to a numeric R vector.

`x` MPCR object.

R matrix converter:

`MPCR.ToNumericMatrix(x)`: Converts an MPCR object to a numeric R matrix.

`x` MPCR object.

Examples

```
# Example usage of the class and its methods
library(MPCR)
a <- matrix(1:36, 6, 6)
MPCR_matrix <- as.MPCR(a, nrow=6, ncol=6, precision="single")
r_vector <- MPCR.ToNumericVector(MPCR_matrix)
r_vector
r_matrix <- MPCR.ToNumericMatrix(MPCR_matrix)
r_matrix
```

Description

Binary arithmetic for numeric/MPCR objects.

Usage

```
## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 + e2

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

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

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 / e2

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 ^ e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 + e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 * e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 - e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 / e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 ^ e2
```

Arguments

`e1, e2` Numeric/MPCR objects.

Value

An MPCR object, matching the data type of the highest precision input.

Examples

```
library(MPCR)
s1 <- as.MPCR(1:20,nrow=2,ncol=10,"single")
s2 <- as.MPCR(21:40,nrow=2,ncol=10,"double")

x <- s1 + s2
typeof(x) # A 64-bit precision (double) MPCR matrix.

s3 <- as.MPCR(1:20,nrow=2,ncol=10,"single")
x <- s1 + s3
typeof(x) # A 32-bit precision (single) MPCR matrix.
```

04-Comparison

Binary comparison operators for numeric/MPCR objects.

Description

Binary comparison operators for numeric/MPCR objects.

Usage

```
## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 < e2

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 <= e2

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

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 != e2

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 > e2

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
e1 >= e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 < e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 <= e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 == e2
```

```
## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 != e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 > e2

## S4 method for signature 'Rcpp_MPCR,BaseLinAlg'
e1 >= e2
```

Arguments

`e1, e2` Numeric/MPCR objects.

Value

A vector/matrix of logicals.

Examples

```
library(MPCR)
s1 <- as.MPCR(1:20, nrow=2, ncol=10, "single")
s2 <- as.MPCR(21:40, nrow=2, ncol=10, "double")

x <- s1 > s2
```

05-Copy

copy

Description

Functions for copying MPCR objects.

Value

An MPCR copy from the input object.

MPCR deep copy

Create a copy of an MPCR object. Typically, using 'equal' creates a new pointer for the object, resulting in any modifications made to object one affecting object two as well.

copy:

`MPCR.copy(x)`: Create a new copy of an MPCR object.

`x` MPCR object.

Examples

```

library(MPCR)
# Example usage of the class and its methods
a <- matrix(1:36, 6, 6)
MPCR_matrix <- as.MPCR(a,nrow=6,ncol=6,precision="single")

# Normal equal '=' will create a new pointer of the object, so any change in object A
# will affect object B
temp_MPCR_matrix = MPCR_matrix
temp_MPCR_matrix[2,2] <- 500
MPCR_matrix[2,2]          #500

MPCR_matrix_copy <- MPCR.copy(MPCR_matrix)
MPCR_matrix[2,2] <-100
MPCR_matrix_copy[2,2] <- 200

MPCR_matrix[2,2]          #100
MPCR_matrix_copy[2,2]     #200

```

06-Dimensions

dimensions

Description

Returns the number of rows or cols in an MPCR object.

Usage

```

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

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

```

Arguments

x An MPCR object.

Value

The number of rows/cols in an MPCR object.

Examples

```
library(MPCR)
x <- as.MPCR(1:16,4,4,"single")
y <- as.MPCR(1:20,4,5,"double")
rows_x <- nrow(x)
cols_y <- ncol(y)
```

07-Extract-Replace *Extract or replace elements from an MPCR object.*

Description

Extract or replace elements from an MPCR object using the '[', '[[', '[<-', and '[[<-' operators. When extracting values, they will be converted to double precision. However, if you update a single object, the double value will be cast down to match the precision. If the MPCR object is a matrix and you access it using the 'i' index, the operation is assumed to be performed in column-major order, or using 'i' and 'j' index.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
x[i, j, drop = TRUE]
## S4 replacement method for signature 'Rcpp_MPCR'
x[i, j, ...] <- value
## S4 method for signature 'Rcpp_MPCR'
x[[i, drop = TRUE]]
## S4 replacement method for signature 'Rcpp_MPCR'
x[[i, ...]] <- value
```

Arguments

x	An MPCR object.
i	Row index or indices.
j	Column index or indices.
...	ignored.
drop	ignored.
value	A value to replace the selected elements with.

Examples

```
library(MPCR)
x <- as.MPCR(1:50,precision="single")
ext <- x[5]
x[5] <- 0
x$ToMatrix(5,10)
x[2,5]
x[3,5] <- 100
```

08-Concatenate	<i>concatenate</i>
----------------	--------------------

Description

`c()` function for MPCR objects.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
MPCR.Concatenate(x)
```

Arguments

`x` List of MPCR objects.

Value

MPCR object containing values from all objects in the list.

Examples

```
library(MPCR)
x <- as.MPCR(1:20,precision="single")
y <- as.MPCR(1:20,precision="single")
list <- c(x,y)
new_obj <- MPCR.Concatenate(list)
```

09-Bind	<i>bind</i>
---------	-------------

Description

`rbind()` and `cbind()` for MPCR objects.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
MPCR.rbind(x,y)

## S4 method for signature 'Rcpp_MPCR'
MPCR.cbind(x,y)
```

Arguments

`x` An MPCR object.
`y` An MPCR object.

Value

An MPCR object, matching the data type of the highest precision input.

Examples

```
library(MPCR)
# create 2 MPCR matrix a,b
a <- as.MPCR(1:20,nrow=2,ncol=10,"single")
b <- as.MPCR(21:40,nrow=2,ncol=10,"double")

x <- MPCR.rbind(a,b)
y <- MPCR.cbind(a,b)
```

10-Diagonal	<i>diag</i>
-------------	-------------

Description

Returns the diagonal of an MPCR matrix.

Usage

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

Arguments

x An MPCR matrix.

Value

An MPCR vector contains the main diagonal of the matrix.

Examples

```
library(MPCR)
x <- as.MPCR(1:16,4,4,"single")
diag_vals <- diag(x)
```

Description

Min-Max functions for MPCR objects values and indices, all NA values are disregarded.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
min(x)

## S4 method for signature 'Rcpp_MPCR'
max(x)

## S4 method for signature 'Rcpp_MPCR'
which.min(x)

## S4 method for signature 'Rcpp_MPCR'
which.max(x)
```

Arguments

`x` An MPCR object.

Value

Min/max value/index.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
min <- min(x)
min_idx <- which.min(x)
```

Description

exp/log functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
exp(x)

## S4 method for signature 'Rcpp_MPCR'
expm1(x)

## S4 method for signature 'Rcpp_MPCR'
log(x, base = 1)

## S4 method for signature 'Rcpp_MPCR'
log10(x)

## S4 method for signature 'Rcpp_MPCR'
log2(x)
```

Arguments

<code>x</code>	An MPCR object.
<code>base</code>	The logarithm base. If <code>base = 1</code> , <code>exp(1)</code> is assumed, only base 1,2, and 10 available.

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
log(x)
```

Description

Finite, infinite, and NaNs.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
is.finite(x)

## S4 method for signature 'Rcpp_MPCR'
```

```
is.infinite(x)

## S4 method for signature 'Rcpp_MPCR'
is.nan(x)
```

Arguments

x An MPCR object.

Value

A bool vector/matrix of the same dimensions as the input.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
is.nan(sqrt(x))
```

14-Miscmath

Miscellaneous mathematical functions

Description

Miscellaneous mathematical functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
abs(x)

## S4 method for signature 'Rcpp_MPCR'
sqrt(x)
```

Arguments

x An MPCR object.

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
sqrt(x)
```

15-NA's	NA's
---------	------

Description

`is.na()`, `na.omit()`, and `na.exclude()` for MPCR objects.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
MPCR.is.na(object,index=-1)
## S4 method for signature 'Rcpp_MPCR'
MPCR.na.exclude(object,value)
## S4 method for signature 'Rcpp_MPCR'
MPCR.na.omit(object)
```

Arguments

<code>object</code>	MPCR object.
<code>index</code>	If a particular index in the MPCR matrix/vector is specified, it will be checked. If no index is provided, all elements will be checked.
<code>value</code>	Value to replace all NAN with.

Value

`MPCR.is.na` will return matrix/vector/bool according to input of the function.
`MPCR.na.exclude` & `MPCR.na.omit` will not return anything.

Examples

```
library(MPCR)
x <- as.MPCR(1:20,precision="single")
x[1] <- NAN
MPCR.is.na(x,index=1) #TRUE
MPCR.na.exclude(x,50)
x[1] #50
```

16-Replicate	<i>replicate</i>
--------------	------------------

Description

Replicates the given input number of times according to count/len , only one should be set at a time, and in case both values are given, only the len value will have effect.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
rep(x, count=0, len=0)
```

Arguments

- x An MPCR object.
- count Value to determine how many times the input value will be replicated.
- len Value to determine the required output size, the input will be replicated until it matches the output len size.

Value

MPCR vector containing the replicated values.

Examples

```
library(MPCR)
x <- as.MPCR(1:16,4,4,"single")
rep_vals_1 <- rep(x,count=2) #output size will be 16*2
rep_vals_2 <- rep(x,len=2) #output size will be 2
```

17-Round	<i>Rounding functions</i>
----------	---------------------------

Description

Rounding functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
ceiling(x)

## S4 method for signature 'Rcpp_MPCR'
floor(x)

## S4 method for signature 'Rcpp_MPCR'
trunc(x)

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

Arguments

`x` An MPCR object.

`digits` The number of digits to use in rounding.

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

input <- runif(20,-1,1)
x <- as.MPCR(input,precision="double")
floor(x)
```

18-Scale

scale

Description

Center or scale an MPCR object.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
scale(x, center, scale)
```

Arguments

`x` An MPCR object.

`center, scale` Logical or MPCR objects.

Value

An MPCR matrix.

Examples

```
library(MPCR)
input <- as.MPCR(1:50, precision="single")
x$ToMatrix(5, 10)
temp_center_scale <- new(1:10, precision="double")
z <- scale(x=input, center=FALSE, scale=temp_center_scale)
```

19-Sweep	<i>sweep</i>
----------	--------------

Description

Sweep an MPCR vector through an MPCR matrix.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
sweep(x, stat, margin, FUN)
```

Arguments

- x An MPCR object.
- stat MPCR vector containing the value(s) that should be used in the operation.
- margin 1 means row; otherwise means column.
- FUN Sweeping function; must be one of "+", "-", "*", "/", or "^".

Value

An MPCR matrix of the same type as the highest precision input.

Examples

```
library(MPCR)
x <- as.MPCR(1:20, 10, 2, "single")
y <- as.MPCR(1:5, precision="double")
sweep_out <- sweep(x, stat=y, margin=1, FUN="+")
MPCR.is.double(sweep_out) #TRUE
```

20-Special Math	<i>Special mathematical functions.</i>
-----------------	----------------------------------------

Description

Special mathematical functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
gamma(x)

## S4 method for signature 'Rcpp_MPCR'
lgamma(x)
```

Arguments

x An MPCR object.

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
lgamma(x)
```

21-Trig	<i>Trigonometric functions</i>
---------	--------------------------------

Description

Basic trig functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
sin(x)

## S4 method for signature 'Rcpp_MPCR'
cos(x)
```

```
## S4 method for signature 'Rcpp_MPCR'
tan(x)

## S4 method for signature 'Rcpp_MPCR'
asin(x)

## S4 method for signature 'Rcpp_MPCR'
acos(x)

## S4 method for signature 'Rcpp_MPCR'
atan(x)
```

Arguments

x An MPCR object.

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

mpcr_matrix <- as.MPCR(1:20,nrow=2,ncol=10,"single")
x <- sin(mpcr_matrix)
```

Description

These functions give the obvious hyperbolic functions. They respectively compute the hyperbolic cosine, sine, tangent, and their inverses, arc-cosine, arc-sine, arc-tangent (or 'area cosine', etc).

Usage

```
## S4 method for signature 'Rcpp_MPCR'
sinh(x)
## S4 method for signature 'Rcpp_MPCR'
cosh(x)
## S4 method for signature 'Rcpp_MPCR'
tanh(x)
## S4 method for signature 'Rcpp_MPCR'
asinh(x)
## S4 method for signature 'Rcpp_MPCR'
acosh(x)
```

```
## S4 method for signature 'Rcpp_MPCR'
atanh(x)
```

Arguments

x An MPCR object.

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

mpcr_matrix <- as.MPCR(1:20,nrow=2,ncol=10,precision="single")
x <- sinh(mpcr_matrix)
```

23-Transpose	<i>transpose</i>
--------------	------------------

Description

Transpose an MPCR object.

Usage

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

Arguments

x An MPCR object.

Value

An MPCR object.

Examples

```
library(MPCR)
a <- matrix(1:20, nrow = 2)
a_MPCR <- as.MPCR(a,2,10,"double")
a_MPCR_transpose <- t(a_MPCR)
```

24-Check precision *Metadata functions*

Description

Checks the precision of a given MPCR object.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
MPCR.is.single(x)
## S4 method for signature 'Rcpp_MPCR'
MPCR.is.half(x)
## S4 method for signature 'Rcpp_MPCR'
MPCR.is.double(x)
## S4 method for signature 'Rcpp_MPCR'
MPCR.is.float(x)
```

Arguments

`x` An MPCR object.

Value

Boolean indicates the precision of the object according to the used function.

Examples

```
library(MPCR)
x <- as.MPCR(1:20,precision="double")
MPCR.is.double(x) #TRUE
MPCR.is.single(x) #FALSE
```

25-Metadata

Metadata functions

Description

Metadata functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
storage.mode(x)
## S4 method for signature 'Rcpp_MPCR'
typeof(x)
## S4 method for signature 'Rcpp_MPCR'
MPCR.object.size(x)
## S4 method for signature 'Rcpp_MPCR'
MPCR.ChangePrecision(x,precision)
```

Arguments

`x` An MPCR object.

`precision` String with the required precision.

Value

Prints/change metadata about an MPCR object.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
typeof(x)
MPCR.ChangePrecision(x,"single")
MPCR.is.single(x) #True
```

26-Print

print

Description

Prints the precision and type of the object, and print will print the meta data of the object without printing the values. Function `x$PrintValues()` should be used to print the values."

Usage

```
## S4 method for signature 'Rcpp_MPCR'
print(x)

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

Arguments

`x`, object An MPCR objects.

Details

Prints metadata about the object and some values.

Value

A string containing the metadata of the MPCR object.

Examples

```
library(MPCR)
x <- as.MPCR(1:16,4,4,"single")
y <- as.MPCR(1:20,4,5,"double")
x
print(y)
```

27-Cholesky decomposition
cholesky decomposition

Description

Performs the Cholesky factorization of a positive definite MPCR matrix `x`.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
chol(x, upper_triangle=TRUE)
```

Arguments

`x` An MPCR matrix.

`upper_triangle` Boolean to check on which triangle the cholesky decomposition should be applied.

Value

An MPCR matrix.

Examples

```
library(MPCR)
x <- as.MPCR(c(1.21, 0.18, 0.13, 0.41, 0.06, 0.23,
              0.18, 0.64, 0.10, -0.16, 0.23, 0.07,
              0.13, 0.10, 0.36, -0.10, 0.03, 0.18,
              0.41, -0.16, -0.10, 1.05, -0.29, -0.08,
              0.06, 0.23, 0.03, -0.29, 1.71, -0.10,
              0.23, 0.07, 0.18, -0.08, -0.10, 0.36),6,6,precision="double")
chol_out <- chol(x)
```

28-Cholesky inverse

cholesky inverse

Description

Performs the inverse of the original matrix using the Cholesky factorization of an MPCR matrix x.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
chol2inv(x, size = NCOL(x))
```

Arguments

x	An MPCR object.
size	The number of columns to use.

Value

An MPCR object.

Examples

```
library(MPCR)
x <- as.MPCR(c(1.21, 0.18, 0.13, 0.41, 0.06, 0.23,
              0.18, 0.64, 0.10, -0.16, 0.23, 0.07,
              0.13, 0.10, 0.36, -0.10, 0.03, 0.18,
              0.41, -0.16, -0.10, 1.05, -0.29, -0.08,
              0.06, 0.23, 0.03, -0.29, 1.71, -0.10,
              0.23, 0.07, 0.18, -0.08, -0.10, 0.36),6,6,precision="single")
chol_out <- chol(x)
chol <- chol2inv(chol_out)
```

29-Crossprod *crossprod*

Description

Calculates the cross product of two MPCR matrices. It uses BLAS routine `gemm()` for $\mathbf{A} \times \mathbf{B}$ operations and `syrk()` for $\mathbf{A} \times \mathbf{A}^T$ operations.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
crossprod(x, y = NULL)

## S4 method for signature 'Rcpp_MPCR'
tcrossprod(x, y = NULL)
```

Arguments

<code>x</code>	An MPCR object.
<code>y</code>	Either <code>NULL</code> , or an MPCR matrix.

Details

Calculates cross product of two MPCR matrices performs:

`x %*% y, t(x) %*% x`

This function uses blas routine `gemm()` for $\mathbf{A} \times \mathbf{B}$ operations & `syrk()` for $\mathbf{A} \times \mathbf{A}^T$ operations.

Value

An MPCR matrix.

Examples

```
library(MPCR)
x <- as.MPCR(1:16,4,4,"single")
y <- as.MPCR(1:20,4,5,"double")

z <- crossprod(x)      # t(x) x
z <- tcrossprod(x)     # x t(x)
z <- crossprod(x,y)    # x y
z <- x %*% y           # x y
```

```
30-Eigen decomposition
      eigen decomposition
```

Description

Solves a system of equations or invert an MPCR matrix, using lapack routine `syevr()`

Usage

```
## S4 method for signature 'Rcpp_MPCR'
eigen(x, only.values = FALSE)
```

Arguments

`x` An MPCR object.
`only.values` (TRUE/FALSE)?

Value

A list contains MPCR objects describing the values and optionally vectors.

Examples

```
library(MPCR)
s <- runif(10, 3)
cross_prod <- crossprod(s)
x <- as.MPCR(cross_prod, nrow(cross_prod), nrow(cross_prod), precision)
y <- eigen(x)
```

```
31-Symmetric              isSymmetric
```

Description

Check if a given MPCR matrix is symmetric.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
isSymmetric(object, ...)
```

Arguments

`object` An MPCR matrix.
`...` Ignored.

Value

A logical value.

Examples

```
library(MPCR)

x <- as.MPCR(1:50,25,2,"Single")
isSymmetric(x) #false

crossprod_output<-crossprod(x)
isSymmetric(crossprod_output) #true
```

32-Norm	<i>norm</i>
---------	-------------

Description

Compute norm.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
norm(x, type = "O")
```

Arguments

- x An MPCR object.
- type "O"-ne, "I"-nfinity, "F"-robenius, "M"-ax modulus, and "l" norms.

Value

An MPCR object.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
norm(x, type="O")
```

33-QR decomposition
QR decomposition

Description

QR factorization and related functions.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
qr(x, tol = 1e-07)

## S4 method for signature 'ANY'
qr.Q(qr, complete = FALSE, Dvec)

## S4 method for signature 'ANY'
qr.R(qr, complete = FALSE)
```

Arguments

<code>x</code>	An MPCR matrix.
<code>qr</code>	QR decomposition MPCR object.
<code>tol</code>	The tolerance for determining numerical column rank.
<code>complete</code>	Should the complete or truncated factor be returned?
<code>Dvec</code>	Vector of diagonals to use when re-constructing Q (default is 1's).

Details

The factorization is performed by the LAPACK routine `geqp3()`. This should be similar to calling `qr()` on an ordinary R matrix with the argument `LAPACK=TRUE`.

Value

`qr` Output of `qr()`.

Examples

```
library(MPCR)

qr_input <- as.MPCR( c(1, 2, 3, 2, 4, 6, 3, 3, 3), 3, 3, "single")
qr_out <- qr(qr_input)
qr_out
qr_out[["qr"]]$PrintValues()
```

```
qr_out[["qraux"]]$PrintValues()
qr_out[["pivot"]]$PrintValues()
qr_out[["rank"]]$PrintValues()

qr_q <- qr.Q(qr_out)
qr_q
```

34-Reciprocal condition

reciprocal condition

Description

Compute matrix norm.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
rcond(x, norm = "O", useInv = FALSE)
```

Arguments

<code>x</code>	An MPCR object.
<code>norm</code>	"O"-ne or "I"-nfinity norm.
<code>useInv</code>	TRUE to use the lower triangle only.

Value

An MPCR Object.

Examples

```
library(MPCR)

x <- as.MPCR(1:20, precision="double")
rcond(x)
```

35-Solve

*solve***Description**

Solve a system of equations or invert an MPCR matrix.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
solve(a, b = NULL, ...)
```

Arguments

<code>a, b</code>	An MPCR objects.
<code>...</code>	Ignored.

Value

Solves the equation $AX=B$.and if $B=NULL$ $t(A)$ will be used.

Examples

```
library(MPCR)

x <- as.MPCR(1:20, 4, 5, "double")
y <- crossprod(x)
solve(y)
```

36-Singular value decomposition
SVD

Description

SVD factorization.

Usage

```
## S4 method for signature 'Rcpp_MPCR'
La.svd(x, nu = min(n, p), nv = min(n, p))

## S4 method for signature 'Rcpp_MPCR'
svd(x, nu = min(n, p), nv = min(n, p))
```

Arguments

`x` An MPCR matrix.

`nu, nv` The number of left/right singular vectors to return.

Details

The factorization is performed by the LAPACK routine `gesdd()`.

Value

The SVD decomposition of the MPCR matrix.

Examples

```
library(MPCR)
svd_vals <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 1, 1, 1)

x <- as.MPCR(svd_vals, 9, 4, "single")
y <- svd(x)
```

37-Back/Forward solve

Back/Forward solve

Description

Solves a system of linear equations where the coefficient matrix is upper or lower triangular. The function solves the equation $A X = B$, where A is the coefficient matrix, X is the solution vector, and B is the right-hand side vector.

Usage

```
## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
backsolve(r, x, k = ncol(r), upper.tri = TRUE, transpose = FALSE)

## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
forwardsolve(l, x, k = ncol(l), upper.tri = FALSE, transpose = FALSE)
```


Arguments

<code>l</code>	An MPCR object.
<code>r</code>	An MPCR object.
<code>x</code>	An MPCR object whose columns give the right-hand sides for the equations.
<code>k</code>	The number of columns of <code>r</code> and rows of <code>x</code> to use.
<code>upper.tri</code>	logical; if TRUE, the upper triangular part of <code>r</code> is used. Otherwise, the lower one.
<code>transpose</code>	logical; if TRUE, solve for $t(l, r) \%*\% \text{output} == x$.

Value

An MPCR object represents the solution to the system of linear equations.

Examples

```
library(MPCR)
a <- matrix(c(2, 0, 0, 3), nrow = 2)
b <- matrix(c(1, 2), nrow = 2)
a_MPCR <- as.MPCR(a, 2, 2, "single")
b_MPCR <- as.MPCR(b, 2, 1, "double")
x <- forwardsolve(a_MPCR, b_MPCR)
x
```

38-MPCR GEMM

MPCR GEMM (Matrix-Matrix Multiplication)

Description

Performs matrix-matrix multiplication of two given MPCR matrices to performs:

$C = \alpha A * B + \beta C$

$C = \alpha A A^T + \beta C$

Usage

```
## S4 method for signature 'Rcpp_MPCR'
MPCR.gemm(a, b = NULL, c, transpose_a= FALSE, transpose_b=FALSE, alpha=1, beta=0)
```

Arguments

<code>a</code>	An MPCR matrix A.
<code>b</code>	An MPCR matrix B, if NULL, the function will perform syrk operation from blas.
<code>c</code>	Input/Output MPCR matrix C.

transpose_a A flag to indicate whether transpose matrix A should be used, if B is NULL and transpose_a =TRUE
The function will perform the following operation:
C=alphaA^TXA+betaC.

transpose_b A flag to indicate whether transpose matrix B should be used.

alpha Specifies the scalar alpha.

beta Specifies the scalar beta.

Value

An MPCR matrix.

Examples

```
library(MPCR)
# create 3 MPCR matrices a,b,c
print(c)
MPCR.gemm(a,b,c,transpose_a=false,transpose_b=TRUE,alpha=1,beta=1)
print(c)
```

39-MPCR TRSM	<i>MPCR TRSM (Triangular Solve)</i>
--------------	-------------------------------------

Description

Solves a triangular matrix equation.
performs:
op(A)*X=alpha*B
X*op(A)=alpha*B

Usage

```
## S4 method for signature 'Rcpp_MPCR'
MPCR.trsm(a,b,upper_triangle,transpose,side = 'L',alpha =1)
```

Arguments

a MPCR Matrix A.

b MPCR Matrix B.

upper_triangle If the value is TRUE, the referenced part of matrix A corresponds to the upper triangle, with the opposite triangle assumed to contain zeros.

transpose If TRUE, the transpose of A is used.

side 'R' for Right side, 'L' for Left side.

alpha Factor used for A, If alpha is zero, A is not accessed.

Value

An MPCR Matrix.

Examples

```
library(MPCR)
a <- matrix(c(3.12393, -1.16854, -0.304408, -2.15901,
             -1.16854, 1.86968, 1.04094, 1.35925,
             -0.304408, 1.04094, 4.43374, 1.21072,
             -2.15901, 1.35925, 1.21072, 5.57265), 4, 4)

mat_a <- as.MPCR(a, 4, 4, "single")
mat_b <- as.MPCR(a, 4, 4, "double")

MPCR.trsm(a=mat_a, b=mat_b, side='R', upper_triangle=TRUE, transpose=FALSE, alpha=1)
print(mat_b)
```

Index

!=, Rcpp_MPCR, BaseLinAlg-method
 (04-Comparison), 6
 !=, Rcpp_MPCR, Rcpp_MPCR-method
 (04-Comparison), 6
 *Topic **S4 class**
 01-MPCR, 2
 *, Rcpp_MPCR, BaseLinAlg-method
 (03-Arithmetic), 5
 *, Rcpp_MPCR, Rcpp_MPCR-method
 (03-Arithmetic), 5
 +, Rcpp_MPCR, BaseLinAlg-method
 (03-Arithmetic), 5
 +, Rcpp_MPCR, Rcpp_MPCR-method
 (03-Arithmetic), 5
 -, Rcpp_MPCR, BaseLinAlg-method
 (03-Arithmetic), 5
 -, Rcpp_MPCR, Rcpp_MPCR-method
 (03-Arithmetic), 5
 /, Rcpp_MPCR, BaseLinAlg-method
 (03-Arithmetic), 5
 /, Rcpp_MPCR, Rcpp_MPCR-method
 (03-Arithmetic), 5
 <, Rcpp_MPCR, BaseLinAlg-method
 (04-Comparison), 6
 <, Rcpp_MPCR, Rcpp_MPCR-method
 (04-Comparison), 6
 <=, Rcpp_MPCR, BaseLinAlg-method
 (04-Comparison), 6
 <=, Rcpp_MPCR, Rcpp_MPCR-method
 (04-Comparison), 6
 ==, Rcpp_MPCR, BaseLinAlg-method
 (04-Comparison), 6
 ==, Rcpp_MPCR, Rcpp_MPCR-method
 (04-Comparison), 6
 >, Rcpp_MPCR, BaseLinAlg-method
 (04-Comparison), 6
 >, Rcpp_MPCR, Rcpp_MPCR-method
 (04-Comparison), 6
 >=, Rcpp_MPCR, BaseLinAlg-method
 (04-Comparison), 6
 >=, Rcpp_MPCR, Rcpp_MPCR-method
 (04-Comparison), 6
 [, Rcpp_MPCR-method
 (07-Extract-Replace), 9
 [<-, Rcpp_MPCR-method
 (07-Extract-Replace), 9
 [[, Rcpp_MPCR-method
 (07-Extract-Replace), 9
 [[<-, Rcpp_MPCR-method
 (07-Extract-Replace), 9
 ^, Rcpp_MPCR, BaseLinAlg-method
 (03-Arithmetic), 5
 ^, Rcpp_MPCR, Rcpp_MPCR-method
 (03-Arithmetic), 5
 01-MPCR, 2
 02-Converters, 4
 03-Arithmetic, 5
 04-Comparison, 6
 05-Copy, 7
 06-Dimensions, 8
 07-Extract-Replace, 9
 08-Concatenate, 10
 09-Bind, 10
 10-Diagonal, 11
 11-Extremes, 12
 12-Log, 12
 13-Mathis, 13
 14-Miscmath, 14
 15-NA's, 15
 16-Replicate, 16
 17-Round, 16
 18-Scale, 17
 19-Sweep, 18
 20-Special Math, 19
 21-Trig, 19
 22-Hyperbolic, 20
 23-Transpose, 21
 24-Check precision, 22

- 25-Metadata, [22](#)
- 26-Print, [23](#)
- 27-Cholesky decomposition, [24](#)
- 28-Cholesky inverse, [25](#)
- 29-Crossprod, [26](#)
- 30-Eigen decomposition, [27](#)
- 31-Symmetric, [27](#)
- 32-Norm, [28](#)
- 33-QR decomposition, [29](#)
- 34-Reciprocal condition, [30](#)
- 35-Solve, [31](#)
- 36-Singular value decomposition, [31](#)
- 37-Back/Forward solve, [32](#)
- 38-MPCR GEMM, [33](#)
- 39-MPCR TRSM, [34](#)
- abs, Rcpp_MPCR-method
(14-Miscmath), [14](#)
- acos, Rcpp_MPCR-method (21-Trig),
[19](#)
- acosh, Rcpp_MPCR-method
(22-Hyperbolic), [20](#)
- arithmetic (03-Arithmetic), [5](#)
- as.MPCR (02-Converters), [4](#)
- asin, Rcpp_MPCR-method (21-Trig),
[19](#)
- asinh, Rcpp_MPCR-method
(22-Hyperbolic), [20](#)
- atan, Rcpp_MPCR-method (21-Trig),
[19](#)
- atanh, Rcpp_MPCR-method
(22-Hyperbolic), [20](#)
- backsolve, Rcpp_MPCR, Rcpp_MPCR-method
(37-Back/Forward solve), [32](#)
- ceiling, Rcpp_MPCR-method
(17-Round), [16](#)
- Check Precision (24-Check
precision), [22](#)
- chol (27-Cholesky decomposition),
[24](#)
- chol, Rcpp_MPCR-method
(27-Cholesky
decomposition), [24](#)
- chol2inv (28-Cholesky inverse), [25](#)
- chol2inv, Rcpp_MPCR-method
(28-Cholesky inverse), [25](#)
- comparison (04-Comparison), [6](#)
- concatenate (08-Concatenate), [10](#)
- Converters (02-Converters), [4](#)
- copy (05-Copy), [7](#)
- cos, Rcpp_MPCR-method (21-Trig), [19](#)
- cosh, Rcpp_MPCR-method
(22-Hyperbolic), [20](#)
- crossprod (29-Crossprod), [26](#)
- crossprod, Rcpp_MPCR-method
(29-Crossprod), [26](#)
- diag (10-Diagonal), [11](#)
- diag, Rcpp_MPCR-method
(10-Diagonal), [11](#)
- dimensions (06-Dimensions), [8](#)
- eigen (30-Eigen decomposition), [27](#)
- eigen, Rcpp_MPCR-method (30-Eigen
decomposition), [27](#)
- exp, Rcpp_MPCR-method (12-Log), [12](#)
- expm1, Rcpp_MPCR-method (12-Log),
[12](#)
- extremes (11-Extremes), [12](#)
- floor, Rcpp_MPCR-method
(17-Round), [16](#)
- forwardsolve, Rcpp_MPCR, Rcpp_MPCR-method
(37-Back/Forward solve), [32](#)
- gamma, Rcpp_MPCR-method
(20-Special Math), [19](#)
- hyperbolic (22-Hyperbolic), [20](#)
- is.finite, Rcpp_MPCR-method
(13-Mathis), [13](#)
- is.infinite, Rcpp_MPCR-method
(13-Mathis), [13](#)
- is.nan, Rcpp_MPCR-method
(13-Mathis), [13](#)
- isSymmetric (31-Symmetric), [27](#)
- isSymmetric, Rcpp_MPCR-method
(31-Symmetric), [27](#)
- La.svd, Rcpp_MPCR-method
(36-Singular value
decomposition), [31](#)
- lgamma, Rcpp_MPCR-method
(20-Special Math), [19](#)
- log (12-Log), [12](#)

- log, Rcpp_MPCR-method (12-Log), 12
- log10, Rcpp_MPCR-method (12-Log), 12
- log2, Rcpp_MPCR-method (12-Log), 12
- mathis (13-Mathis), 13
- max, Rcpp_MPCR-method (11-Extremes), 12
- metadata (25-Metadata), 22
- min, Rcpp_MPCR-method (11-Extremes), 12
- miscmath (14-Miscmath), 14
- MPCR (01-MPCR), 2
- MPCR_GEMM (38-MPCR_GEMM), 33
- MPCR_TRSM (39-MPCR_TRSM), 34
- MPCR.abs (14-Miscmath), 14
- MPCR.acos (21-Trig), 19
- MPCR.acosh (22-Hyperbolic), 20
- MPCR.Add (03-Arithmetic), 5
- MPCR.asin (21-Trig), 19
- MPCR.asinh (22-Hyperbolic), 20
- MPCR.atan (21-Trig), 19
- MPCR.atanh (22-Hyperbolic), 20
- MPCR.backsolve (37-Back/Forward solve), 32
- MPCR.cbind (09-Bind), 10
- MPCR.cbind, Rcpp_MPCR-method (09-Bind), 10
- MPCR.ceiling (17-Round), 16
- MPCR.ChangePrecision (25-Metadata), 22
- MPCR.ChangePrecision, Rcpp_MPCR-method (25-Metadata), 22
- MPCR.chol (27-Cholesky decomposition), 24
- MPCR.chol2inv (28-Cholesky inverse), 25
- MPCR.Concatenate (08-Concatenate), 10
- MPCR.Concatenate, Rcpp_MPCR-method (08-Concatenate), 10
- MPCR.copy (05-Copy), 7
- MPCR.cos (21-Trig), 19
- MPCR.cosh (22-Hyperbolic), 20
- MPCR.crossprod (29-Crossprod), 26
- MPCR.diag (10-Diagonal), 11
- MPCR.Divide (03-Arithmetic), 5
- MPCR.eigen (30-Eigen decomposition), 27
- MPCR.exp (12-Log), 12
- MPCR.expml (12-Log), 12
- MPCR.floor (17-Round), 16
- MPCR.forwardsolve (37-Back/Forward solve), 32
- MPCR.gamma (20-Special Math), 19
- MPCR.gemm (38-MPCR_GEMM), 33
- MPCR.gemm, Rcpp_MPCR-method (38-MPCR_GEMM), 33
- MPCR.is.double (24-Check precision), 22
- MPCR.is.double, Rcpp_MPCR-method (24-Check precision), 22
- MPCR.is.finite (13-Mathis), 13
- MPCR.is.float (24-Check precision), 22
- MPCR.is.float, Rcpp_MPCR-method (24-Check precision), 22
- MPCR.is.half (24-Check precision), 22
- MPCR.is.half, Rcpp_MPCR-method (24-Check precision), 22
- MPCR.is.infinite (13-Mathis), 13
- MPCR.is.na (15-NA's), 15
- MPCR.is.na, Rcpp_MPCR-method (15-NA's), 15
- MPCR.is.nan (13-Mathis), 13
- MPCR.is.single (24-Check precision), 22
- MPCR.is.single, Rcpp_MPCR-method (24-Check precision), 22
- MPCR.isSymmetric (31-Symmetric), 27
- MPCR.La.svd (36-Singular value decomposition), 31
- MPCR.lgamma (20-Special Math), 19
- MPCR.log (12-Log), 12
- MPCR.log10 (12-Log), 12
- MPCR.log2 (12-Log), 12
- MPCR.max (11-Extremes), 12
- MPCR.min (11-Extremes), 12
- MPCR.Multiply (03-Arithmetic), 5
- MPCR.na.exclude (15-NA's), 15
- MPCR.na.exclude, Rcpp_MPCR-method (15-NA's), 15
- MPCR.na.omit (15-NA's), 15
- MPCR.na.omit, Rcpp_MPCR-method (15-NA's), 15

- MPCR.ncol (*06-Dimensions*), 8
- MPCR.norm (*32-Norm*), 28
- MPCR.nrow (*06-Dimensions*), 8
- MPCR.object.size (*25-Metadata*), 22
- MPCR.object.size, Rcpp_MPCR-method
(*25-Metadata*), 22
- MPCR.Power (*03-Arithmetic*), 5
- MPCR.print (*26-Print*), 23
- MPCR.qr (*33-QR decomposition*), 29
- MPCR.rbind (*09-Bind*), 10
- MPCR.rbind, Rcpp_MPCR-method
(*09-Bind*), 10
- MPCR.rcond (*34-Reciprocal condition*), 30
- MPCR.rep (*16-Replicate*), 16
- MPCR.round (*17-Round*), 16
- MPCR.scale (*18-Scale*), 17
- MPCR.show (*26-Print*), 23
- MPCR.sin (*21-Trig*), 19
- MPCR.sinh (*22-Hyperbolic*), 20
- MPCR.solve (*35-Solve*), 31
- MPCR.sqrt (*14-Miscmath*), 14
- MPCR.storage.mode (*25-Metadata*), 22
- MPCR.str (*26-Print*), 23
- MPCR.Subtract (*03-Arithmetic*), 5
- MPCR.svd (*36-Singular value decomposition*), 31
- MPCR.sweep (*19-Sweep*), 18
- MPCR.t (*23-Transpose*), 21
- MPCR.tan (*21-Trig*), 19
- MPCR.tanh (*22-Hyperbolic*), 20
- MPCR.tcrossprod (*29-Crossprod*), 26
- MPCR.ToNumericMatrix
(*02-Converters*), 4
- MPCR.ToNumericVector
(*02-Converters*), 4
- MPCR.trsm (*39-MPCR TRSM*), 34
- MPCR.trsm, Rcpp_MPCR-method
(*39-MPCR TRSM*), 34
- MPCR.trunc (*17-Round*), 16
- MPCR.typeof (*25-Metadata*), 22
- MPCR.which.max (*11-Extremes*), 12
- MPCR.which.min (*11-Extremes*), 12
- NA's (*15-NA's*), 15
- ncol, Rcpp_MPCR-method
(*06-Dimensions*), 8
- new, 3
- norm (*32-Norm*), 28
- norm, Rcpp_MPCR-method (*32-Norm*), 28
- nrow, Rcpp_MPCR-method
(*06-Dimensions*), 8
- print (*26-Print*), 23
- print, Rcpp_MPCR-method
(*26-Print*), 23
- qr (*33-QR decomposition*), 29
- qr, Rcpp_MPCR-method (*33-QR decomposition*), 29
- qr.Q, ANY-method (*33-QR decomposition*), 29
- qr.R, ANY-method (*33-QR decomposition*), 29
- rcond (*34-Reciprocal condition*), 30
- rcond, Rcpp_MPCR-method
(*34-Reciprocal condition*), 30
- Rcpp_MPCR-class (*01-MPCR*), 2
- rep, Rcpp_MPCR-method
(*16-Replicate*), 16
- replicate (*16-Replicate*), 16
- round (*17-Round*), 16
- round, Rcpp_MPCR-method
(*17-Round*), 16
- scale (*18-Scale*), 17
- scale, Rcpp_MPCR-method
(*18-Scale*), 17
- show, Rcpp_MPCR-method (*26-Print*), 23
- sin, Rcpp_MPCR-method (*21-Trig*), 19
- sinh, Rcpp_MPCR-method
(*22-Hyperbolic*), 20
- solve (*35-Solve*), 31
- solve, Rcpp_MPCR-method
(*35-Solve*), 31
- specialmath (*20-Special Math*), 19
- sqrt, Rcpp_MPCR-method
(*14-Miscmath*), 14
- storage.mode, Rcpp_MPCR-method
(*25-Metadata*), 22
- svd (*36-Singular value decomposition*), 31

`svd`, Rcpp_MPCR-method
 (36-Singular value decomposition), 31

`sweep` (19-Sweep), 18

`sweep`, Rcpp_MPCR-method
 (19-Sweep), 18

`t`, Rcpp_MPCR-method
 (23-Transpose), 21

`tan`, Rcpp_MPCR-method (21-Trig), 19

`tanh`, Rcpp_MPCR-method
 (22-Hyperbolic), 20

`tcrossprod`, Rcpp_MPCR-method
 (29-Crossprod), 26

`transpose` (23-Transpose), 21

`trig` (21-Trig), 19

`trunc`, Rcpp_MPCR-method
 (17-Round), 16

`typeof`, Rcpp_MPCR-method
 (25-Metadata), 22

`which.max`, Rcpp_MPCR-method
 (11-Extremes), 12

`which.min`, Rcpp_MPCR-method
 (11-Extremes), 12