Package 'MPCR'

November 16, 2023

Type Package

Title Multi Precision Computing

Version 1.0

Date 2023-11-12

Author David Helmy <david.helmy@brightskiesinc.com>
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 with the ability to perform mixed-precision operations through a newly defined tile-based data structure. 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. 'MPCR' also introduces support for the tile-based matrix data structure with mixed precision, enabling the utilization of a range of tile-based linear algebra algorithms.

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

VignetteBuilder utils

Suggests utils

42

Index

R topics documented:

01-MPCR	3
02-MPCRTile	4
03-Converters	6
04-Arithmetic	7
05-Comparison	8
06-Extract-Replace	0
07-Dimensions	1
08-Copy	1
09-Concatenate	2
10-Bind	3
11-Diagonal	4
12-Extremes	4
13-Log	5
14-Mathis	6
15-Misemath	7
16-NA's	7
17-Replicate	8
18-Round	9
19-Scale	20
20-Sweep	20
21-Special Math	21
22-Trig	
23-Hyperbolic	
24-Transpose	
25-Check precision	
26-Metadata	
27-Print	26
28-Cholesky decomposition	27
29-Cholesky inverse	27
30-Crossprod	28
31-Eigen decomposition	29
32-Symmetric	0
33-Norm	
34-QR decomposition	1
35-Reciprocal condition	
36-Solve	3
37-Singular value decomposition	
38-Back/Forward solve	
39-MPCR GEMM	86
40-MPCR TRSM	57
41-MPCRTile GEMM	8
	39
43-MPCRTile TRSM	

01-MPCR 3

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.

4 02-MPCRTile

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()</pre>
```

02-MPCRTile

MPCRTile S4 Class

Description

MPCRTile is a data structure for tile matrices with mixed precision, where each tile possesses a specific precision level.

Value

MPCRTile object (constructor - accessors - methods)

Constructor

```
new creates a new instance of Tile-Matrix MPCRTile class.
new (MPCRTile, rows, cols, rows_per_tile, cols_per_tile, values, precisions)
rows Number of rows in the matrix.
cols Number of cols in the matrix.
rows_per_tile Number of rows in each tile.
cols_per_tile Number of cols in each tile.
values R matrix or vector containing all the values that should be in the matrix.
precisions R matrix or vector of strings, containing precision type of each tile.
```

Accessors

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

Size Total number of elements inside the Matrix.

Row Number of rows.

02-MPCRTile 5

```
Col Number of cols.
```

TileRow Number of rows in each tile.

TileCol Number of cols in each tile.

TileSize Total number of elements in each tile.

Methods

The following methods are available for objects of class MPCRTile:

PrintTile:

PrintTile(tile_row_idx, tile_col_idx): Prints all the values stored inside a specific tile plus meta-data about the tile.

```
tile_row_idx Row index of the tile.
tile col idx Col index of the tile.
```

ChangeTilePrecision:

ChangeTilePrecision(tile_row_idx, tile_col_idx, precision): Change the precision of specific tile, this function will need to copy all the values to cast them to the new precision.

```
tile_row_idx Row index of the tile.
tile_col_idx Col index of the tile.
precision Required new precision as a string.
```

FillSquareTriangle:

FillSquareTriangle (value, upper.tri, precision): Fills upper or lower triangle with a given value and precision, new tiles will be created, replacing the old tiles. **Note:** The input must be a square matrix

value A value used during matrix filling.

upper.tri A flag to indicate what triangle to fill. if TRUE, the upper triangle will be filled, otherwise the lower triangle.

precision The precision of the tiles created during matrix filling, in case it's not a diagonal tile.

Sum: Sum (): Get the sum of all elements in all tiles in MPCRTile Matrix.

Prod: Prod(): Get the product of all elements in all tiles in MPCRTile Matrix.

6 03-Converters

```
"single")

tile_mat <- new(MPCRTile, 6, 6, 2, 2, a, b)
tile_mat
sum <- tile_mat$Sum()
prod <- tile_mat$Prod()
tile_mat$PrintTile(1,1)
tile_mat$ChangeTilePrecision(1,1,"single")

n_rows <- tile_mat$Row
n_cols <- tile_mat$Col
total_size <- tile_mat$Size
rows_per_tile <- tile_mat$TileRow
cols_per_tile <- tile_mat$TileCol</pre>
```

03-Converters

Converters

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:

```
\label{eq:mpcr} \begin{split} & \texttt{MPCR.ToNumericVector}\left(\mathtt{x}\right) \text{: Converts an MPCR object to a numeric } R \text{ vector.} \\ & \texttt{x MPCR object.} \end{split}
```

04-Arithmetic 7

R matrix converter:

 $\label{eq:mpcr} \mbox{MPCR.ToNumericMatrix}(\mbox{x}): \mbox{Converts an MPCR object to a numeric R matrix.} \\ \mbox{x MPCR object.}$

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</pre>
```

04-Arithmetic

Binary arithmetic numeric/MPCR objects.

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,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
```

8 05-Comparison

```
## 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.</pre>
```

05-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</pre>
## S4 method for signature 'Rcpp_MPCR,Rcpp_MPCR'
```

05-Comparison 9

```
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'
## 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.

```
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
```

10 06-Extract-Replace

06-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</pre>
```

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.
```

```
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</pre>
```

07-Dimensions

07-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)</pre>
```

08-Сору

copy

Description

Functions for copying MPCR objects.

Value

An MPCR copy from the input object.

12 09-Concatenate

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.
```

MPCRTile deep copy

Create a duplicate of an MPCRTile object. Usually, using 'equal' creates a new pointer for the object, causing any modifications made to object one to affect object two as well.

copy:

```
MPCRTile.copy(x): Create a new copy of an MPCRTile matrix. x MPCRTile matrix.
```

Examples

```
library (MPCR)
# Example usage of the class and its methods
a \leftarrow matrix(1:36, 6, 6)
MPCR_matrix <- as.MPCR(a, nrow=6, ncol=6, precision="single")</pre>
# 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)</pre>
MPCR_matrix[2,2] < -100
MPCR_matrix_copy[2,2] <- 200
MPCR_matrix[2,2]
                            #100
                             #200
MPCR_matrix_copy[2,2]
```

09-Concatenate

concatenate

Description

c () function for MPCR objects.

Usage

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

10-Bind 13

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)</pre>
```

10-Bind

bind

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.

```
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)</pre>
```

14 12-Extremes

11-Diagonal

diag

Description

Returns the diagonal of an MPCR matrix.

Usage

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

Arguments

Х

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)</pre>
```

12-Extremes

Min-Max Functions

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)
```

13-Log

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)</pre>
```

13-Log

Logarithms and Exponentials

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

```
x An MPCR object.
```

base The logarithm base. If base = 1, exp(1) is assumed, only base 1,2, and 10 avail-

able.

14-Mathis

Value

An MPCR object of the same dimensions as the input.

Examples

```
library(MPCR)

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

14-Mathis

Finite, infinite, and NaNs

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

Х

An MPCR object.

Value

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

```
library(MPCR)

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

15-Miscmath

15-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

Х

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)</pre>
```

16-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)
```

17-Replicate

Arguments

object MPCR object.

index 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.

value 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</pre>
```

17-Replicate

replicate

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.

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.

18-Round 19

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</pre>
```

18-Round

Rounding functions

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

 ${\tt x}$ An MPCR object. digits The number of digits to use in rounding.

Value

An MPCR object of the same dimensions as the input.

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

20-Sweep

19-Scale

scale

Description

Center or scale an MPCR object.

Usage

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

Arguments

```
{\tt 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)</pre>
```

20-Sweep

sweep

Description

Sweep an MPCR vector through an MPCR matrix.

Usage

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

21-Special Math

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.
FIIN	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</pre>
```

21-Special Math

Special mathematical functions.

Description

Special mathematical functions.

Usage

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

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

Arguments

Х

An MPCR object.

Value

An MPCR object of the same dimensions as the input.

```
library(MPCR)

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

22-Trig

22-Trig

Trigonometric functions

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

Х

An MPCR object.

Value

An MPCR object of the same dimensions as the input.

```
library(MPCR)

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

23-Hyperbolic 23

23-Hyperbolic Hyperbolic functions

Description

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

```
library(MPCR)

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

24 25-Check precision

24-Transpose

transpose

Description

Transpose an MPCR object.

Usage

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

Arguments

Х

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)</pre>
```

25-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)
```

26-Metadata 25

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</pre>
```

26-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.

26 27-Print

Examples

```
library(MPCR)

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

27-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.

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

```
28-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

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)
```

```
29-Cholesky inverse

cholesky inverse
```

Description

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

28 30-Crossprod

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

30-Crossprod

crossprod

Description

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

Usage

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

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

Arguments

```
x An MPCR object.
```

y Either NULL, or an MPCR matrix.

29

Details

```
Calculates cross product of two MPCR matrices performs: x \%*\% y, t(x) \%*\% x
This function uses blas routine gemm() for A \times B operations & syrk() for A \times 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</pre>
```

```
31-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.

32-Symmetric

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)</pre>
```

32-Symmetric

is Symmetric

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.

```
library(MPCR)

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

crossprod_output<-crossprod(x)
isSymmetric(crossprod_output) #true</pre>
```

33-Norm 31

33-Norm norm

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 "1" norms.
```

Value

An MPCR object.

Examples

```
library(MPCR)

x <- as.MPCR(1:20,precision="double")
norm(x, type="0")</pre>
```

```
34-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

X	An MPCR matrix.
qr	QR decomposition MPCR object.
tol	The tolerance for determining numerical column rank.
complete	Should the complete or truncated factor be returned?
Dvec	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

```
gr Output of gr().
```

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()</pre>
qr_out[["rank"]]$PrintValues()
```

```
35-Reciprocal condition reciprocal\ condition
```

Description

Compute matrix norm.

Usage

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

36-Solve 33

Arguments

x An MPCR object.

norm "O"-ne or "I"-nfinity norm.

useInv TRUE to use the lower triangle only.

Value

An MPCR Object.

Examples

```
library(MPCR)

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

36-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

a, b An MPCR objects.

... Ignored.

Value

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

```
library(MPCR)

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

```
37-Singular value decomposition \ensuremath{\textit{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.

38-Back/Forward solve 35

```
38-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

1	An MPCR object.
r	An MPCR object.
Х	An MPCR object whose columns give the right-hand sides for the equations.
k	The number of columns of r and rows of x to use.
upper.tri	logical; if TRUE, the upper triangular part of r is used. Otherwise, the lower one.
transpose	logical; if TRUE, solve for $t(1, r)$ %*% output == x.

Value

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

```
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</pre>
```

36 39-MPCR GEMM

39-MPCR GEMM MA

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

a	An MPCR matrix A.
b	An MPCR matrix B, if NULL, the function will perform syrk operation from

blas.

c 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.

```
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)
```

40-MPCR TRSM 37

40-MPCR TRSM

MPCR TRSM (Triangular Solve)

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

Value

An MPCR Matrix.

38 41-MPCRTile GEMM

```
41-MPCRTile GEMM MPCRTile GEMM (Matrix-Matrix Multiplication)
```

Description

Tile-based matrix-matrix multiplication of two given MPCR tiled matrices to **perform:** $C = alpha*A \times B + beta*C$

Usage

```
## S4 method for signature 'Rcpp_MPCRTile'
MPCRTile.gemm(a,b,c,transpose_a= FALSE,transpose_b=FALSE,alpha=1,beta=0,num_threads
```

Arguments

a An MPCR tile matrix A.
b An MPCR tile matrix B.

c Input/Output MPCR tile matrix C.

transpose_a A flag to indicate whether transpose matrix A should be used.

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

alpha Specifies the scalar alpha.
beta Specifies the scalar beta.

num_threads An integer to determine number if thread to run using openmp, default = 1 (serial

with no parallelization).

Value

An MPCR tile matrix C.

```
library(MPCR)
# create 3 MPCR Tile matrices a,b,c
a <- as.MPCR(1:20,5,4,"single")
b <- as.MPCR(1:20,4,5,"single")
c <- as.MPCR(1:16,4,4,"single")
print(c)
MPCRTile.gemm(a,b,c,transpose_a=false,transpose_b=TRUE,alpha=1,beta=1,num_threads = 8)
print(c)</pre>
```

42-MPCRTile POTRF 39

```
42-MPCRTile POTRF MPCRTile Chol (Cholesky decomposition)
```

Description

Tile-based Cholesky decomposition of a positive definite tile-based symmetric matrix.

Usage

```
## S4 method for signature 'Rcpp_MPCRTile'
chol(x, overwrite_input = TRUE, num_threads = 1)
```

Arguments

Value

An MPCR tile matrix.

```
library (MPCR)
 a \leftarrow matrix(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
b <- c("float", "double", "float", "float",
      "double", "double", "float", "float",
      "double")
 chol_mat <- new(MPCRTile, 6, 6, 2, 2, a, b)
 x <- chol(chol_mat,overwrite_input=FALSE,num_threads=8)</pre>
 print(chol_mat)
 print(x)
 chol(chol_mat)
 print(chol_mat)
```

40 43-MPCRTile TRSM

```
43-MPCRTile TRSM MPCRTile TRSM (Triangular Solve)
```

Description

```
Tile-based algorithm to solve a triangular matrix equation for MPCR tiled matrices. performs: op(A)*X=alpha*B X*op(A)=alpha*B
```

Usage

```
## S4 method for signature 'Rcpp_MPCRTile'
MPCRTile.trsm(a,b,side,upper_triangle,transpose,alpha)
```

Arguments

```
a An MPCR tile matrix A.

b An MPCR tile matrix B, X after returning.

side 'R' for right side, 'L' for left side.

upper_triangle

What part of the matrix A is referenced (if TRUE upper triangle is referenced), the opposite triangle being assumed to be zero.

transpose

If TRUE, the transpose of A is used.

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

Value

```
An MPCR Tile Matrix B \rightarrow (X).
```

43-MPCRTile TRSM 41

print(mat_b)

Index

!=,Rcpp_MPCR,BaseLinAlg-method (05-Comparison),8	>=, Rcpp_MPCR, BaseLinAlg-method (05-Comparison), 8
!=,Rcpp_MPCR,Rcpp_MPCR-method	>=, Rcpp_MPCR, Rcpp_MPCR-method
(05-Comparison), 8	(05-Comparison), 8
*Topic S4 class	[,Rcpp_MPCR-method
01-MPCR, 3	(06-Extract-Replace), 10
02-MPCRTile,4	[<-,Rcpp_MPCR-method
	(06-Extract-Replace), 10
*,Rcpp_MPCR,BaseLinAlg-method (04-Arithmetic),7	[[,Rcpp_MPCR-method
	(06-Extract-Replace), 10
*, Rcpp_MPCR, Rcpp_MPCR-method	[[<-,Rcpp_MPCR-method
(04-Arithmetic), 7	(06-Extract-Replace), 10
+, Rcpp_MPCR, BaseLinAlg-method (04-Arithmetic), 7	^, Rcpp_MPCR, BaseLinAlg-method
	(04-Arithmetic),7
+, Rcpp_MPCR, Rcpp_MPCR-method (04-Arithmetic), 7	^, Rcpp_MPCR, Rcpp_MPCR-method
-, Rcpp_MPCR, BaseLinAlg-method	(04-Arithmetic), 7
(04-Arithmetic), 7	01-MPCR, 3
-, Rcpp_MPCR, Rcpp_MPCR-method	02-MPCRTile,4
(04-Arithmetic), 7	03-Converters, 6
/,Rcpp_MPCR,BaseLinAlg-method	04-Arithmetic, 7
(04-Arithmetic), 7	05-Comparison, 8
/,Rcpp_MPCR,Rcpp_MPCR-method	06-Extract-Replace, 10
(04-Arithmetic), 7	07-Dimensions, 11
<pre><, Rcpp_MPCR, BaseLinAlg-method</pre>	08-Copy, 11
(05-Comparison), 8	09-Concatenate, 12
<, Rcpp_MPCR, Rcpp_MPCR-method	10-Bind, 13
(05-Comparison), 8	11-Diagonal, 14
<=, Rcpp_MPCR, BaseLinAlg-method	12-Extremes, 14
(05-Comparison), 8	13-Log, 15
<=, Rcpp_MPCR, Rcpp_MPCR-method	14-Mathis, 16
(05-Comparison), 8	15-Miscmath, 17
==, Rcpp_MPCR, BaseLinAlg-method	16-NA's, 17
(05-Comparison), 8	17-Replicate, 18
==, Rcpp_MPCR, Rcpp_MPCR-method	18-Round, 19
(05-Comparison), 8	19-Scale, 20
>, Rcpp_MPCR, BaseLinAlg-method	20-Sweep, 20
(05-Comparison), 8	21-Special Math, 21
>, Rcpp_MPCR, Rcpp_MPCR-method	22-Trig, <mark>22</mark>
(05-Comparison), 8	23-Hyperbolic, 23
· • • • • • • • • • • • • • • • • • • •	- · · · · · · · · · · · · · · · · · · ·

24-Transpose, 24	chol, Rcpp_MPCR-method
25-Check precision, 24	(28-Cholesky
26-Metadata, 25	decomposition), 27
27-Print, 26	chol, Rcpp_MPCRTile-method
28-Cholesky decomposition, 27	(42-MPCRTile POTRF), 39
29-Cholesky inverse, 27	chol2inv(29-Cholesky inverse), 27
30-Crossprod, 28	chol2inv, Rcpp_MPCR-method
31-Eigen decomposition, 29	(29-Cholesky inverse), 27
32-Symmetric, 30	comparison (05-Comparison), 8
33-Norm, 31	concatenate (09-Concatenate), 12
34-QR decomposition, 31	Converters (03-Converters), 6
35-Reciprocal condition, 32	copy (08-Copy), 11
36-Solve, 33	cos, Rcpp_MPCR-method (22-Trig), 22
37-Singular value decomposition,	cosh, Rcpp_MPCR-method
34	(23-Hyperbolic), 23
38-Back/Forward solve, 35	crossprod (30-Crossprod), 28
39-MPCR GEMM, 36	crossprod, Rcpp_MPCR-method
40-MPCR TRSM, 37	(30-Crossprod), 28
41-MPCRTile GEMM, 38	(• • • • • • • • • • • • • • • • • • •
42-MPCRTile POTRF, 39	diag(11-Diagonal), 14
	diag, Rcpp_MPCR-method
43-MPCRTile TRSM, 40	(11-Diagonal), 14
1 2 200	dimensions (07-Dimensions), 11
abs, Rcpp_MPCR-method	
(15-Miscmath), 17	eigen (31-Eigen decomposition), 29
acos, Rcpp_MPCR-method(22-Trig),	eigen, Rcpp_MPCR-method (31-Eigen
22	decomposition), 29
acosh, Rcpp_MPCR-method	exp, Rcpp_MPCR-method (13-Log), 15
(23-Hyperbolic), 23	expm1, Rcpp_MPCR-method (13-Log),
arithmetic(04-Arithmetic),7	15
as.MPCR(03-Converters),6	extremes (12-Extremes), 14
asin, Rcpp_MPCR-method(22-Trig),	
22	floor, Rcpp_MPCR-method
asinh, Rcpp_MPCR-method	(18-Round), 19
(23-Hyperbolic), 23	forwardsolve, Rcpp_MPCR, Rcpp_MPCR-method
atan, Rcpp_MPCR-method(22-Trig), 22	(38-Back/Forward solve), 35
atanh, Rcpp_MPCR-method	gamma, Rcpp_MPCR-method
(23-Hyperbolic), <mark>23</mark>	(21-Special Math), 21
	(21-Special mach), 21
backsolve, Rcpp_MPCR, Rcpp_MPCR-method	hyperbolic (23-Hyperbolic), 23
(38-Back/Forward solve), 35	
	is.finite,Rcpp_MPCR-method
ceiling, Rcpp_MPCR-method	(14-Mathis), 16
(18-Round), 19	is.infinite, Rcpp_MPCR-method
Check Precision (25-Check	(14-Mathis), 16
precision), 24	is.nan, Rcpp_MPCR-method
chol (28-Cholesky decomposition),	(14-Mathis), 16
27	isSymmetric(32-Symmetric), 30

's Comment of a Decree MDCD model and	MDCD Consists and Done MDCD matheway
isSymmetric, Rcpp_MPCR-method	MPCR.Concatenate, Rcpp_MPCR-method
(32-Symmetric), 30	(09-Concatenate), 12
	MPCR.copy (08-Copy), 11
La.svd, Rcpp_MPCR-method	MPCR.cos (22-Trig), 22
(37-Singular value	MPCR.cosh (23-Hyperbolic), 23
decomposition), 34	MPCR.crossprod (30-Crossprod), 28
lgamma, Rcpp_MPCR-method	MPCR.diag(11-Diagonal), 14
(21-Special Math), 21	MPCR.Divide (04-Arithmetic), 7
log (13-Log), 15	MPCR.eigen (31-Eigen
log, Rcpp_MPCR-method(13-Log), 15	decomposition), 29
$log10$, Rcpp_MPCR-method(13-Log),	MPCR.exp $(13-Log)$, 15
15	MPCR.expm1 (13-Log), 15
$log2$, Rcpp_MPCR-method (13-Log), 15	MPCR.floor (18-Round), 19
	MPCR.forwardsolve
mathis (14-Mathis), 16	(38-Back/Forward solve),35
max, Rcpp_MPCR-method	MPCR.gamma (21-Special Math), 21
(12-Extremes), 14	MPCR.gemm (39-MPCR GEMM), 36
metadata (26-Metadata), 25	MPCR.gemm, Rcpp_MPCR-method
min, Rcpp_MPCR-method	(39-MPCR GEMM), 36
(12-Extremes), 14	MPCR.is.double(25-Check
miscmath (15-Miscmath), 17	precision), 24
MPCR (01-MPCR), 3	MPCR.is.double, Rcpp_MPCR-method
MPCR GEMM (39-MPCR GEMM), 36	(25-Check precision), 24
MPCR TRSM (40-MPCR TRSM), 37	MPCR.is.finite(14-Mathis), 16
MPCR.abs (15-Miscmath), 17	MPCR.is.float (25-Check
MPCR.acos (22-Trig), 22	precision), 24
MPCR.acosh (23-Hyperbolic), 23	MPCR.is.float, Rcpp_MPCR-method
MPCR.Add(04-Arithmetic), 7	(25-Check precision), 24
MPCR.asin (22-Trig), 22	MPCR.is.half (25-Check
MPCR.asinh (23-Hyperbolic), 23	precision), 24
MPCR.atan (22-Trig), 22	MPCR.is.half, Rcpp_MPCR-method
MPCR.atanh (23-Hyperbolic), 23	(25-Check precision), 24
MPCR.backsolve(38-Back/Forward	MPCR.is.infinite (14-Mathis), 16
solve), 35	MPCR.is.na (16-NA's), 17
	MPCR.is.na, Rcpp_MPCR-method
MPCR.cbind(10-Bind), 13	(16-NA's), 17
MPCR.cbind, Rcpp_MPCR-method	MPCR.is.nan (14-Mathis), 16
(10-Bind), 13	
MPCR.ceiling (18-Round), 19	MPCR.is.single (25-Check
MPCR.ChangePrecision	precision), 24
(26-Metadata), 25	MPCR.is.single, Rcpp_MPCR-method
MPCR.ChangePrecision, Rcpp_MPCR-method	
(26-Metadata), 25	MPCR.isSymmetric(32-Symmetric),
MPCR.chol (28-Cholesky	30
decomposition), 27	MPCR.La.svd(37-Singular value
MPCR.chol2inv(29-Cholesky	decomposition), 34
inverse), 27	MPCR.lgamma (21-Special Math), 21
MPCR.Concatenate	MPCR.log(13-Log), 15
(09-Concatenate), 12	MPCR.log10 (13-Log), 15

MPCR.log2 $(13-Log)$, 15	MPCR.trsm,Rcpp_MPCR-method
MPCR.max(12-Extremes), 14	(40-MPCR TRSM), 37
MPCR.min (12-Extremes), 14	MPCR.trunc (18-Round), 19
MPCR.Multiply(04-Arithmetic),7	MPCR.typeof(26-Metadata),25
MPCR.na.exclude (16-NA's), 17	MPCR.which.max(12-Extremes), 14
MPCR.na.exclude, Rcpp_MPCR-method	MPCR.which.min(12-Extremes), 14
(16-NA's), 17	MPCRTile(02-MPCRTile),4
MPCR.na.omit (16-NA's), 17	MPCRTile Chol(42-MPCRTile
MPCR.na.omit, Rcpp_MPCR-method	POTRF), 39
(16-NA's), 17	MPCRTile GEMM (41-MPCRTile GEMM),
MPCR.ncol(07-Dimensions), 11	38
MPCR.norm (33-Norm), 31	MPCRTile.chol(42-MPCRTile
MPCR.nrow(07-Dimensions), 11	POTRF), 39
MPCR.object.size(26-Metadata), 25	MPCRTile.copy(08-Copy), 11
MPCR.object.size, Rcpp_MPCR-method	MPCRTile.gemm (41 -MPCRTile GEMM),
(26-Metadata), <mark>25</mark>	38
MPCR.Power(04-Arithmetic),7	MPCRTile.gemm, Rcpp_MPCRTile-method
MPCR.print (27-Print), 26	(41-MPCRTile GEMM), 38
MPCR.qr(34-QR decomposition), 31	MPCRTile.trsm(43-MPCRTile TRSM),
MPCR.rbind(10-Bind), 13	40
MPCR.rbind, Rcpp_MPCR-method	MPCRTile.trsm, Rcpp_MPCRTile-method
(10-Bind), 13	(43-MPCRTile TRSM),40
MPCR.rcond(35-Reciprocal	NA's (16-NA's), 17
condition), 32	ncol, Rcpp_MPCR-method
MPCR.rep(17-Replicate), 18	(07-Dimensions), 11
MPCR.round (18-Round), 19	new, 3, 4
MPCR.scale(19-Scale), 20	norm (33-Norm), 31
MPCR.show $(27-Print)$, 26	norm, Rcpp_MPCR-method (33-Norm),
MPCR.sin(22-Trig), 22	31
MPCR.sinh(23-Hyperbolic), 23	nrow, Rcpp_MPCR-method
MPCR.solve (36-Solve), 33	(07-Dimensions), 11
MPCR.sqrt (15-Miscmath), 17	(6,7,21,10,110,110), 11
MPCR.storage.mode(26-Metadata),	print (27-Print), <mark>26</mark>
25	print,Rcpp_MPCR-method
MPCR.str(27-Print), 26	(27-Print), <mark>26</mark>
MPCR.Subtract(04-Arithmetic),7	(24.00.)
MPCR.svd(37-Singular value	qr(34-QR decomposition), 31
decomposition), 34	qr, Rcpp_MPCR-method (34-QR
MPCR.sweep $(20-Sweep)$, 20	decomposition), 31
MPCR.t (24-Transpose), 24	qr.Q, ANY-method (34-QR
MPCR.tan (22-Trig), 22	decomposition), 31
MPCR.tanh (23-Hyperbolic), 23	qr.R, ANY-method (34-QR
MPCR.tcrossprod(30-Crossprod), 28	decomposition), 31
MPCR.ToNumericMatrix	rcond(35-Reciprocal condition),
(03-Converters), 6	32
MPCR.ToNumericVector	rcond, Rcpp_MPCR-method
(03-Converters), 6	(35-Reciprocal condition),
MPCR.trsm(40-MPCR TRSM).37	32

Rcpp_MPCR-class(01-MPCR),3
Rcpp_MPCRTile-class
(02-MPCRTile), 4
rep,Rcpp_MPCR-method
(17-Replicate), 18
replicate(17-Replicate),18
round (18-Round), 19
round, Rcpp_MPCR-method
(18-Round), 19
(10 110 and), 15
scale(19-Scale),20
scale, Rcpp_MPCR-method
(19-Scale), 20
show, Rcpp_MPCR-method(27-Print),
26
sin, Rcpp_MPCR-method (22-Trig), 22
sinh,Rcpp_MPCR-method
(23-Hyperbolic), 23
solve(36-Solve), 33
solve,Rcpp_MPCR-method
(36-Solve), <mark>33</mark>
specialmath(21-Special Math),21
sqrt,Rcpp_MPCR-method
(15-Miscmath), 17
storage.mode,Rcpp_MPCR-method
(26-Metadata), <mark>25</mark>
svd(37-Singular value
decomposition), 34
svd, Rcpp_MPCR-method
(37-Singular value
decomposition), 34
sweep $(20-Sweep)$, $\frac{20}{20}$
sweep, Rcpp_MPCR-method
(20-Sweep), 20
$(20^{\circ} \text{Sweep}), 20^{\circ}$
t Dans MDCD mathad
t, Rcpp_MPCR-method
(24-Transpose), 24
tan, Rcpp_MPCR-method(22-Trig), 22
tanh, Rcpp_MPCR-method
(23-Hyperbolic), 23
tcrossprod,Rcpp_MPCR-method
(30-Crossprod), 28
transpose (24-Transpose), <mark>24</mark>
trig <i>(22-Trig)</i> , <mark>22</mark>
trunc,Rcpp_MPCR-method
(18-Round), <mark>19</mark>
typeof, Rcpp_MPCR-method
(26-Metadata), <mark>25</mark>

which.max,Rcpp_MPCR-method
 (12-Extremes), 14
which.min,Rcpp_MPCR-method
 (12-Extremes), 14