Package 'rmumps'

November 3, 2026

Type Package

Title Wrapper for 'MUMPS' Library

Version 5.9.1-1

Maintainer Balasubramanian Narasimhan <naras@stanford.edu>

Description Some basic features of 'MUMPS' (Multifrontal Massively Parallel sparse direct Solver) are wrapped in a class whose methods can be used for sequentially solving a sparse linear system (symmetric or not) with one or many right hand sides (dense or sparse). There is a possibility to do separately symbolic analysis, LU (or LDL^t) factorization and system solving. The ordering libraries 'PORD' and 'METIS' are included. 'MUMPS' method was first described in Amestoy et al. (2001) <doi:10.1137/S0895479899358194> and Amestoy et al. (2006) <doi:10.1016/j.parco.2005.07.004>.

License GPL (>= 2)

Copyright file inst/COPYRIGHTS

Depends R (>= 3.5.0), methods

Imports Rcpp (>= 0.12.0)

LinkingTo Rcpp

SystemRequirements GNU Make

NeedsCompilation yes

Biarch yes

Suggests testthat, Matrix, slam

BugReports https://github.com/bnaras/rmumps/issues

URL https://mumps-solver.org/, https://github.com/bnaras/rmumps/

Encoding UTF-8

Config/roxygen2/version 8.1.0

Author Serguei Sokol [aut],
Balasubramanian Narasimhan [ctb, cph, cre] (C interface for other packages, ABI adapters, vendoring pipeline),

Emmanuel Agullo [ctb] (MUMPS),
Patrick Amestoy [ctb, cph] (MUMPS, AMD),
Maurice Bremond [ctb] (MUMPS),
Alfredo Buttari [ctb] (MUMPS),
Philippe Combes [ctb] (MUMPS),
Marie Durand [ctb] (MUMPS),
Aurelia Fevre [ctb] (MUMPS),
Matthieu Gerest [ctb] (MUMPS),
Abdou Guermouche [ctb] (MUMPS),
Antoine Jego [ctb] (MUMPS),
Guillaume Joslin [ctb] (MUMPS),
Jacko Koster [ctb] (MUMPS),
Jean-Yves L'Excellent [ctb] (MUMPS),
Theo Mary [ctb] (MUMPS),
Pierre-Etienne Polet [ctb] (MUMPS),
Stephane Pralet [ctb] (MUMPS),
Chiara Puglisi [ctb] (MUMPS),
Francois-Henry Rouet [ctb] (MUMPS),
Wissam Sid-Lakhdar [ctb] (MUMPS),
Tzvetomila Slavova [ctb] (MUMPS),
Bora Ucar [ctb] (MUMPS),
Clement Weisbecker [ctb] (MUMPS),
CERFACS [cph] (MUMPS),
CNRS [cph] (MUMPS, rmumps),
ENS Lyon [cph] (MUMPS),
INP Toulouse [cph] (MUMPS),
Inria [cph] (MUMPS),
Mumps Technologies [cph] (MUMPS),
University of Bordeaux [cph] (MUMPS),
Timothy A. Davis [ctb, cph] (AMD, in MUMPS),
Iain S. Duff [ctb, cph] (AMD, in MUMPS),
The University of Tennessee and The University of Tennessee Research Foundation [cph] (LAPACK-derived code in MUMPS),
The University of California Berkeley [cph] (LAPACK-derived code in MUMPS),
The University of Colorado Denver [cph] (LAPACK-derived code in MUMPS),
Juergen Schulze [ctb] (PORD (public domain)),
George Karypis [ctb] (METIS),
Regents of the University of Minnesota [cph] (METIS),
Douglas C. Schmidt [ctb] (glibc qsort, in METIS/GKlib),
Free Software Foundation, Inc. [cph] (glibc code, in METIS/GKlib),
Alexander Chemeris [ctb, cph] (msinttypes, in METIS/GKlib),
Makoto Matsumoto [ctb, cph] (MT19937-64, in METIS/GKlib),
Takuji Nishimura [ctb, cph] (MT19937-64, in METIS/GKlib),
INRAE [cph] (rmumps),
INSA [cph] (rmumps)

Repository CRAN

Date/Publication 2026-11-03 09:20:11 UTC

Contents

rmumps-package . . . 3
Rcpp_Rmumps-class . . . 4
RMUMPS_PERM . . . 6
Rmumps__del_ptr . . . 7
Rmumps__get_permutation . . . 8
Rmumps__pointer_api . . . 8
Rmumps__ptr_ijv . . . 10
Rmumps__set_mat_ptr . . . 10
Rmumps__set_permutation . . . 11
Rmumps__solveptr . . . 12
Rmumps__triplet . . . 12
solvet . . . 13

Index **14**

`rmumps-package` *Rcpp port of MUMPS library for LU or LDL^t factorization of sparse matrices*

Description

Creates a MUMPS compatible object storing a sparse matrix. Gives a possibility to do separately symbolic analysis, factorization and system solving.

Details

Create a new Rmumps object with `A <- Rmumps$new(asparse)` then solve a linear system with one or many right hand sides `x <- solve(A, b)`. Cf. Rmumps

Author(s)

Serguei Sokol, INRA

Maintainer: Serguei Sokol (sokol at insa-toulouse.fr)

References

MUMPS official site https://mumps-solver.org

Sokol S. *rmumps: Wrapper for 'MUMPS' Library*. https://CRAN.R-project.org/package=rmumps

Examples

```
# 3 x 3 system given as triplets (0-based indices): A = diag(4, 5, 6)
A <- Rmumps$new(0:2, 0:2, c(4, 5, 6), 3L)
x <- solve(A, c(4, 10, 18))
x  # 1 2 3
```

Rcpp_Rmumps-class *Rcpp Exported Class Wrapping MUMPS library*

Description

This class can be used for storing sparse matrix and solving corresponding linear system with one or many right hand sides. There is a possibility to do separately symbolic analysis, LU factorization and system solving.

Fields

`sym`: integer (read only), 0=non symmetric matrix, 1=symmetric with pivots on diagonal or 2=general symmetric

`copy`: logical, copy or not rhs and matrix values

`mrhs`: numeric matrix, multiple rhs (always overwritten with solution)

`rhs`: numeric vector, single rhs (always overwritten with solution)

Methods

`new(asp, sym=0, copy=TRUE)`: constructor from Matrix::dgTMatrix class (or from convertible to it) and slam::simple_triplet_matrix class

`new(i, j, x, n, copy=TRUE)`: constructor from triade rows, cols, vals

`symbolic()`: do symbolic analysis (stored internally)

`numeric()`: do LU or LDL^t factorization (stored internally)

`solve(b)`: solve single rhs (if b is a vector) or multiple rhs if b is a matrix (can be dense or sparse). Return the solution(s).

`solvet(b)`: same as solve() but solves with transposed matrix

`det()`: Return determinant of the matrix

`inv()`: Return inverse of the matrix)

`set_mat_data(x)`: updates matrix entries (x must be in the same order as in previous calls

`set_icntl(iv, ii)`: set ICNTL parameter vector

`get_icntl()`: get ICNTL parameter vector

`set_cntl(v, iv)`: set CNTL parameter vector

`get_cntl()`: get CNTL parameter vector

`get_infos()`: get a named list of information vectors: info, rinfo, infog and rinfog

`dim()`: Return a dimension vector of the matrix

`nrow()`: Return a row number of the matrix

`ncol()`: Return a column number of the matrix

`print()`: Print summary information on the matrix

`show()`: Print summary information on the matrix

`set_keep()`: Set KEEP array elements (undocumented feature of MUMPS)

`get_keep()`: Get a copy of KEEP array elements (length=500)

`set_permutation(perm)`: Set permutation type which can impact storage and factorization performances. Parameter perm can take one of the following predefined integer values RMUMPS_PERM_AMD, RMUMPS_PERM_AMF, RMUMPS_PERM_SCOTCH, RMUMPS_PERM_PORD, RMUMPS_PERM_METIS, RMUMPS_PERM_QAMD. This method should be called once and before symbolic analysis of the matrix. If it is called afterward, a new symbolic and numeric factorization will be performed when one of other methods (e.g. solve()) will request them. In other words, previous symbolic and numeric factorizations are canceled by this method.

`get_permutation()`: get permutation type currently set in the object

`mumps_version()`: Return a string with MUMPS version used in rmumps

Note

When creating a symmetric matrix (sym=1 or sym=2), the upper (or lower) mart of the input matrix must be zeroed.

For meaning of entries in MUMPS vectors cntl, icntl, info, rinfo, infog and rinfog cf. original documentation of MUMPS project.

No need to call symbolic() and numeric() methods before a solve() call.

If in constructor, a parameter `copy` is set to FALSE, no rhs neither matrix copying is done. The solution is written "in place" thus overwriting rhs (watch out side effects)

For a detailed error diagnostic (e.g. when factorizing a singular matrix), use method get_infos() and cf. MUMPS documentation on the official MUMPS site).

Author(s)

Serguei Sokol, INRA

References

MUMPS official site https://mumps-solver.org

Sokol S. *rmumps: Wrapper for 'MUMPS' Library*. https://CRAN.R-project.org/package=rmumps

Examples

```
if (requireNamespace("Matrix", quietly = TRUE)) {
 # prepare random sparse matrix
 library(Matrix)
 library(rmumps)
 n=300
 a=Matrix(0, n, n)
 set.seed(7)
 ij=sample(1:(n*n), 15*n)
 a[ij]=runif(ij)
 diag(a)=0
 diag(a)=-rowSums(a)
```

```
a[1,1]=a[1,1]-1
am=Rmumps$new(a)
b=as.double(a%*%(1:n)) # rhs for an exact solution vector 1:n
# following time includes symbolic analysis, LU factorization and system solving
system.time(x<-solve(am, b))
bb=2*b
# this second time should be much shorter
# as symbolic analysis and LU factorization are already done
system.time(xx<-solve(am, bb))
# compare to Matrix corresponding times
system.time(xm<-solve(a, b))
system.time(xxm<-solve(a, bb))
# compare to Matrix precision
range(x-1:n)  # mumps
range(xm-1:n) # Matrix

# matrix inversion
system.time(aminv <- solve(am))
system.time(ainv <- solve(a)) # the same in Matrix

# symmetric matrix
asy=as(a+t(a), "symmetricMatrix")
bs=as.double(asy%*%(1:n)) # rhs for 1:n solution
au=asy
# Here, we keep only diagonal and upper values of asy matrix.
# It could be also diagonal and lower values.
au[row(au)>col(au)]=0
ams=Rmumps$new(au, sym=1)
system.time(xs<-solve(ams, bs)) # rmumps
system.time(xsm<-solve(asy, bs))# Matrix
# compare to Matrix precision
range(xs-1:n)  # mumps
range(xsm-1:n) # Matrix

# clean up by hand to avoid possible interference between gc() and
# Rcpp object destructor after unloading this namespace
rm(am, ams)
gc()
}
```

RMUMPS_PERM *Exported Constants*

Description

Integer constants defining permutation types and exported from rmumps are following:

- RMUMPS_PERM_AMD
- RMUMPS_PERM_AMF
- RMUMPS_PERM_SCOTCH

- RMUMPS_PERM_PORD
- RMUMPS_PERM_METIS
- RMUMPS_PERM_QAMD
- RMUMPS_PERM_AUTO

They are all regrouped in a named vector RMUMPS_PERM where names are items above and values are corresponding constants.

Examples

```
if (requireNamespace("slam", quietly=TRUE)) {
    am=rmumps::Rmumps$new(slam::as.simple_triplet_matrix(diag(1:3)))
    am$set_permutation(RMUMPS_PERM_AMD)
    am$solve(1:3)
}
```

Rmumps__del_ptr *Delete via Pointer*

Description

This is a C wrapper to Rmumps::~Rmumps() destructor. Available in R too. In C++ code can be used as rmumps::Rmumps__del_ptr(pm)

Usage

```
Rmumps__del_ptr(pm)
```

Arguments

pm	pointer of type XPtr<Rmumps>, object to be deleted

Value

No return value, called for its side effect: the object and its MUMPS instance are freed; pm must not be used afterwards.

Rmumps__get_permutation *Get Permutation Parameter*

Description

This is a C wrapper to `Rmumps::get_permutation()` method. Available in R too. In C++ code can be used as `rmumps::Rmumps__get_permutation(pm)`

Usage

```
Rmumps__get_permutation(pm)
```

Arguments

`pm`	pointer of type XPtr<Rmumps>, object having sparse matrix permuted according to some method.

Value

integer defining permutation method used before matrix decomposition.

Rmumps__pointer_api *Pointer Interface to Rmumps Methods*

Description

C wrappers to methods of the `Rmumps` class, for objects held as pointers (e.g. created with `Rmumps__ptr_ijv` or obtained with `Rmumps__xptr`). Available in R too. In C++ code they can be used as `rmumps::Rmumps__symbolic(pm)` etc. (`LinkingTo: rmumps, Rcpp`).

Usage

```
Rmumps__xptr(obj)

Rmumps__symbolic(pm)

Rmumps__numeric(pm)

Rmumps__solvetptr(pm, pb, lrhs, nrhs)

Rmumps__set_icntl(pm, iv, ii)

Rmumps__get_icntl(pm)

Rmumps__set_cntl(pm, v, iv)
```

```
Rmumps__get_cntl(pm)

Rmumps__get_infos(pm)

Rmumps__det(pm)

Rmumps__log2det(pm)

Rmumps__get_sym_perm(pm)

Rmumps__get_uns_perm(pm)

Rmumps__set_perm_in(pm, p_in)

Rmumps__mumps_version(pm)
```

Arguments

obj	an object created with Rmumps$new().
pm	pointer of type XPtr<Rmumps>.
pb	pointer of type XPtr<double>, vector or dense matrix of rhs, overwritten with the solution.
lrhs	integer, leading dimension in pb.
nrhs	integer, number of rhs to solve.
iv, ii	integer vectors of values and 1-based positions.
v	numeric vector of values.
p_in	integer vector, a permutation of 1:n.

Value

Rmumps__xptr: a pointer of type XPtr<Rmumps> to the object inside obj, without a finalizer (obj keeps owning it and must stay alive). Rmumps__get_icntl, Rmumps__get_cntl, Rmumps__get_infos, Rmumps__det, Rmumps__log2det, Rmumps__get_sym_perm, Rmumps__get_uns_perm, Rmumps__mumps_version: the same values as the corresponding methods of Rmumps. The other functions are called for their side effects and return NULL invisibly.

Examples

```
a <- Rmumps$new(c(0L, 1L), c(0L, 1L), c(2, 4), 2L)
pm <- Rmumps__xptr(a)
Rmumps__numeric(pm)
Rmumps__det(pm)
```

Rmumps__ptr_ijv *Construct via Triplet Pointers*

Description

This is a C wrapper to `Rmumps::Rmumps(i, j, v, n, nz, sym)` constructor. Available in R too. In C++ code can be used as `rmumps::Rmumps__ptr_ijv(pi, pj, pa, n, nz, sym)`

Usage

```
Rmumps__ptr_ijv(pi, pj, pa, n, nz, sym)
```

Arguments

`pi`	pointer of type XPtr<int>, vector of i-indeces for sparse triplet
`pj`	pointer of type XPtr<int>, vector of j-indeces for sparse triplet
`pa`	pointer of type XPtr<double>, vector or values for sparse triplet
`n`	integer, size of the matrix (n x n)
`nz`	integer, number of non zeros in the matrix
`sym`	integer, 0 means general (non symmetric) matrix, 1 - symmetric with pivotes on the main diagonal, 2 - general symmetric (pivotes may be anywhere)

Value

pointer of type XPtr<Rmumps> pointing to newly created object. To avoid memory leakage, it is user's responsibility to call `Rmumps__del_ptr(pm)` in a due moment (where `pm` is the returned pointer).

Rmumps__set_mat_ptr *Set Matrix via Pointer*

Description

This is a C wrapper to `Rmumps::set_mat_ptr(a)` method. Available in R too. In C++ code can be used as `rmumps::Rmumps__set_mat_ptr(pm)`. Using this method invalidates previous numeric decomposition (but not symbolic one).

Usage

```
Rmumps__set_mat_ptr(pm, pa)
```

Arguments

pm	pointer of type XPtr<Rmumps>, object having sparse matrix to be replaced with second parameter
pa	pointer of type XPtr<double>, value vector from sparse triplet providing a new matrix. Structure of the new matrix must be identical to the old one. That's why there is no need to provide i and j for the new triplet.

Value

No return value, called for its side effect: the object now holds the new matrix values (the next solve refactorizes).

Rmumps__set_permutation

Set Permutation Parameter

Description

This is a C wrapper to `Rmumps::set_permutation(permutation)` method. Available in R too. In C++ code can be used as `rmumps::Rmumps__set_permutation(pm, permutation)`

Usage

```
Rmumps__set_permutation(pm, permutation)
```

Arguments

pm	pointer of type XPtr<Rmumps>, object having sparse matrix permuted according to a chosen method.
permutation	integer one of predefined constants (cf. RMUMPS_PERM). Setting a new permutation invalidates current symbolic and numeric matrix decompositions.

Value

No return value, called for its side effect: the ordering method (ICNTL(7)) of the object is set.

Rmumps__solveptr *Solve via Pointer*

Description

This is a C wrapper to `Rmumps::solveptr()` method. Available in R too. In C++ code can be used as `rmumps::Rmumps__solveptr(pobj, pb, lrhs, nrhs)`

Usage

```
Rmumps__solveptr(pobj, pb, lrhs, nrhs)
```

Arguments

`pobj`	pointer of type XPtr<Rmumps>, object having sparse matrix
`pb`	pointer of type XPtr<double>, vector or dense matrix of rhs
`lrhs`	integer, leading dimension in pb
`nrhs`	integer, number of rhs to solve.

Value

No return value, called for its side effect: the right hand sides pointed to by `pb` are overwritten by the solutions.

Rmumps__triplet *Explore via Triplet*

Description

This is a C wrapper to `Rmumps::triplet()` method. Available in R too. In C++ code can be used as `rmumps::Rmumps__triplet(pm)`

Usage

```
Rmumps__triplet(pm)
```

Arguments

`pm`	pointer of type XPtr<Rmumps>, object having sparse matrix to be explored

Value

a list with sparse triplet described with fields i, j, v

`solvet` *Solve a Transposed System*

Description

Solves $A^T x = b$ for an `Rmumps` object holding A.

Usage

```
solvet(a, b, ...)
```

Arguments

`a`	an object created with `Rmumps$new()`.
`b`	right-hand side(s), as accepted by the `solve()` method of `Rmumps`. If missing, the transpose of the inverse is returned.
`...`	not used.

Value

The solution: a vector or a matrix, like `solve()`.

Examples

```
a <- Rmumps$new(c(0L, 1L, 0L), c(0L, 1L, 1L), c(2, 4, 1), 2L)
solvet(a, c(2, 9))
```

Index

∗ **classes**
    Rcpp_Rmumps-class, 4

determinant.Rcpp_Rmumps
        *(*Rcpp_Rmumps-class*)*, 4
dim.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*, 4

ncol.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*, 4
nrow.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*, 4

print.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*, 4

Rcpp_Rmumps-class, 4
Rmumps, *3*
Rmumps *(*Rcpp_Rmumps-class*)*, 4
rmumps *(*rmumps-package*)*, 3
rmumps-package, 3
Rmumps__del_ptr, 7
Rmumps__det *(*Rmumps__pointer_api*)*, 8
Rmumps__get_cntl *(*Rmumps__pointer_api*)*,
        8
Rmumps__get_icntl
        *(*Rmumps__pointer_api*)*, 8
Rmumps__get_infos
        *(*Rmumps__pointer_api*)*, 8
Rmumps__get_permutation, 8
Rmumps__get_sym_perm
        *(*Rmumps__pointer_api*)*, 8
Rmumps__get_uns_perm
        *(*Rmumps__pointer_api*)*, 8
Rmumps__log2det *(*Rmumps__pointer_api*)*, 8
Rmumps__mumps_version
        *(*Rmumps__pointer_api*)*, 8
Rmumps__numeric *(*Rmumps__pointer_api*)*, 8
Rmumps__pointer_api, 8
Rmumps__ptr_ijv, *8*, 10
Rmumps__set_cntl *(*Rmumps__pointer_api*)*,
        8
Rmumps__set_icntl
        *(*Rmumps__pointer_api*)*, 8
Rmumps__set_mat_ptr, 10
Rmumps__set_perm_in
        *(*Rmumps__pointer_api*)*, 8
Rmumps__set_permutation, 11
Rmumps__solveptr, 12
Rmumps__solvetptr
        *(*Rmumps__pointer_api*)*, 8
Rmumps__symbolic *(*Rmumps__pointer_api*)*,
        8
Rmumps__triplet, 12
Rmumps__xptr *(*Rmumps__pointer_api*)*, 8
RMUMPS_PERM, 6, *11*
RMUMPS_PERM_AMD *(*RMUMPS_PERM*)*, 6
RMUMPS_PERM_AMF *(*RMUMPS_PERM*)*, 6
RMUMPS_PERM_AUTO *(*RMUMPS_PERM*)*, 6
RMUMPS_PERM_METIS *(*RMUMPS_PERM*)*, 6
RMUMPS_PERM_PORD *(*RMUMPS_PERM*)*, 6
RMUMPS_PERM_QAMD *(*RMUMPS_PERM*)*, 6
RMUMPS_PERM_SCOTCH *(*RMUMPS_PERM*)*, 6

show.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*, 4
solve.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*, 4
solvet, 13
solvet.Rcpp_Rmumps *(*Rcpp_Rmumps-class*)*,
        4