Package 'umfpackr'

December 5, 2019

Type Package

Title Sparse Linear Algebra with UMFPACK

Version 0.5

Author Rob van Harrevelt [aut, cre]

Maintainer Rob van Harrevelt <rvanharrevelt@gmail.com>

Description Solve linear and non-linear
systems of equations using the sparse linear algebra package UMFPACK.

License GPL-3

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports Rcpp

Depends methods, Matrix

RoxygenNote 7.0.1

Roxygen list(markdown = TRUE, old_usage = TRUE)

Suggests testthat, nleqslv, stringr

NeedsCompilation yes

SystemRequirements
For Linux and other non-Windows platforms the libsuitesparse-dev library is required.

R topics documented:

umf_control *Argument* umf_control

Description

Functions umf_solve and umf_solve_nl have an argument umf_control that can be used to specify UMFPACK control parameters. These control parameters are described in detail in the UMFPACK User Guide. This guide describes how the control options can be specified in C code. In the following paragraphs is explained how the C code can be translated to an R expression passed to argument umf_control.

In C code, the control parameters are stored in a numeric array control with elements specified with named numerical constants. For example, element UMFPACK_ORDERING contains a control parameter that specifies the ordering method. The allowed values for this parameters are specified with constants such as UMFPACK_ORDERING_AMD, UMFPACK_ORDERING_CHOLMOD and UMFPACK_ORDERING_METIS. For example, to use METIS, the specification in C code is:

```
control[UMFPACK_ORDERING] = UMFPACK_ORDERING_METIS;
```

The corresponding R expression passed to argument umf_control is

```
list(ORDERING = "METIS")
```

Thus in the R expression the control names are without prefix UMFPACK_ and the possible values are specified with a character giving the name of the constant in UMFPACK excluding the prefix UMFPACK_ORDERING_.

Some UMFPACK control parameters are specified with numerical values. For example,

```
list(SYM_PIVOT_TOLERANCE = 0.05)
```

specifies the value of the symmetric pivot tolerance (the corresponding C code would be control[UMFPACK_SYM_PIVOT_TOLE = 0.05)

Another example of an R expression for argument umf_control:

```
list(STRATEGY = "UNSYMMETRIC", ORDERING = "METIS",
     SCALE = "NONE")
```

Warning

On Windows, umfpackr does not use the complete SuiteSparse package but onbly the UMFPACK and AMD modules. Therefore the only allowed ordering option on Windows is AMD.

See Also

umf_solve and umf_solve_nl

`umf_solve` *Solves the system of linear equations using UMFPACK*

Description

This function solved the linear equations of the form $Ax = b$ using UMFPACK.

Usage

```
umf_solve(a, b, umf_control = list())
```

Arguments

`a`	an object of class `dgCMatrix` (see `dgCMatrix-class`)
`b`	the vector b
`umf_control`	A named list with control parameters passed to UMFPACK. For example, to use METIS column ordering use `umf_control = list(ORDERING = "METIS")`. Using METIS column ordering may require less memory than ordering ordering options, which may be useful for very large matrices. See for more information `umf_control`.

Value

the solution x

References

Dennis, J.E. Jr and Schnabel, R.B. (1997), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Siam.

Davis, T.A. (2004). A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, **30(2)**, 165–195.

Davis, T.A (2004). Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, **30(2)**, 196–199.

Davis, T.A and Duff, I.S. (1997). An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. Matrix Anal. Applic.*, **18(1)**, 140–158.

Davis, T.A and Duff, I.S (1999). A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, **25(1)**, 1–19..

See Also

`umf_solve_nl` and `umf_control`.

`umf_solve_nl` *Solves a system of non-linear equations using UMFPACK*

Description

This function solves a system of non-linear equations $F(x) = 0$ using Newton's method. UMFPACK is employed to solve the linear equations in each Newton iteration. Optionally a cubic line search is used when a Newton step does not yield a sufficient reduction of the function values.

Usage

```
umf_solve_nl(start, fn, jac, ..., control, global = c("no", "cline"),
  scaling = c("row", "col", "none"), umf_control = list())
```

Arguments

`start`	initial guess of the solution x
`fn`	the function F
`jac`	a function returning the Jacobian of the function as a `dgCMatrix` object
`...`	arguments passed to `fn` and `jac`
`control`	a list with control parameters. See Details.
`global`	The global strategy. Possible values are `"no"` (no global strategy, the default) and `"cline"` (cubic line search) (cubic line search)
`scaling`	Scaling method. Possible values are `"row"`. `"col"` and `"none"`. The default is `"row"`. See Details.
`umf_control`	A named list with control parameters passed to UMFPACK. For example, to use METIS column ordering use `umf_control = list(ORDERING = "METIS")`. Using METIS column ordering may require less memory than ordering ordering options, which may be useful for very large matrices. See for more information `umf_control`.

Details

Control options: Argument `control` is a named list containing one or more of the following components:

`ftol` The function value tolerance. Convergence is reached if the largest function value is smaller than `ftol`. The default value is `1e-8`.

`xtol` The relative step size tolerance. When the relative step size is smaller than `xtol`, then the iteration is stopped. The default value is `1e-8`.

`maxiter` The maximum number of iterations. The default is 20.

`trace` A logical. If `TRUE` then the progress of the iteraton is printed. The default is `FALSE`.

`silent` A logical. If `TRUE` then all output is suppressed. The default is `FALSE`.

`allow_singular` A logical value (default `FALSE`) indicating if a small correction to the Jacobian is applied when it is singular or too ill-conditioned. The method used is similar to a Levenberg-Marquardt correction and is explained in Dennis and Schnabel (1996) on page 151.

`allow_singular` A logical value (default `FALSE`) indicating if a small correction to the Jacobian is applied when it is singular. The method used is similar to a Levenberg-Marquardt correction and is explained in Dennis and Schnabel (1996) on page 151.

Scaling of the Jacobian: For each iteration in the Newton method the linear system $Js = F(x)$ is solved, where the Jacobiab matrix J are the derivatives of the equations with respect to the variables, and s the Newton step. Scaling can improve the condition of the Jacobian. For *row scaling*, the system is transformed to $D^{-1}Js = D^{-1}F(x)$, where D is a diagonal matrix with row scaling factors. Here the scaling factors are the L1 norms of the rows of J. For *column scaling*, the system is transformed to $JD^{-1}Ds = F(x)$, where D is a diagonal matrix with column scaling factors, calculated from the L1 norms of the columns of J.

The scaling is only used to solve the non-linear equations and has no effect on the convergence of the Newton algorihtm. Thus the iterations are considered to be converged when the maximum value of the unscaled function values $F(x)$ is smaller than `ftol`.

Value

a list with the following components:

`solved`	A logical equal to `TRUE` if convergence of the function values has been achieved.
`iter`	the number of iterations
`x`	the final values of x
`fval`	the function value
`message`	A string equal to `"ok"` if a solution has been found. Otherwise it describes the reason why the iteration was stopped without success

References

Dennis, J.E. Jr and Schnabel, R.B. (1997), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Siam.

Davis, T.A. (2004). A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, **30(2)**, 165–195.

Davis, T.A (2004). Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, **30(2)**, 196–199.

Davis, T.A and Duff, I.S. (1997). An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. Matrix Anal. Applic.*, **18(1)**, 140–158.

Davis, T.A and Duff, I.S (1999). A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, **25(1)**, 1–19..

See Also

`umf_solve` and `umf_control`.

Examples

```
library(umfpackr)

dslnex <- function(x, c) {
   y <- numeric(2)
   y[1] <- x[1]^2 + x[2]^2 - c
   y[2] <- exp(x[1]-1) + x[2]^3 - c
   y
}

jacdsln <- function(x, c) {
   n <- length(x)
   Df <- matrix(numeric(n*n),n,n)
   Df[1,1] <- 2*x[1]
   Df[1,2] <- 2*x[2]
   Df[2,1] <- exp(x[1]-1)
   Df[2,2] <- 3*x[2]^2

   return(as(Df, "dgCMatrix"))
}


xstart <- c(2,3)
print(umf_solve_nl(xstart, dslnex, jacdsln, c = 2,
                   control = list(trace = TRUE)))

# now use METIs columns ordering (run this on Linux only)
if (.Platform$OS.type != "windows") {
   print(umf_solve_nl(xstart, dslnex, jacdsln, c = 2,
                   control = list(trace = TRUE),
                   umf_control = list(SCALE = NONE,
                                      ORDERING = "METIS")))
}
```

Index