UMFPACK Interface for R

Rob van Harrevelt

2021-09-16

Introduction

Package `umfpackr` provides an interface to UMFPACK, a sparse direct solver written in C. The package `umfpackr` contains the functions `umf_solve` and `umf_solve_nl` for solving linear and non-linear systems of equations, respectively. This vignette describes the UMPFACK interface implemented in `umfpackr` in detail.

System dependencies

UMFPACK is part of SuiteSparse, a suite of libraries for sparse linear algebra.

On Linux and other non-Windows platforms, `umfpackr` assumes that a full installation of SuiteSparse is available. On Ubuntu installation is very easy:

```
sudo apt-get install libsuitesparse-dev
```

On macOS, installation is also easy with package manager HomeBrew (HomeBrew is not installed by default, so you may have to install HomeBrew first):

```
brew install suite-sparse
```

If it not possible to install SuiteSparse with a package manager on your platform, you have to build SuiteSparse from source.

Because it is difficult to install the full SuiteSparse suite on Windows, `umfpackr` includes the source of the UMFPACK and AMD libraries of the SuiteSparse suite. These libraries are sufficient to solve sparse linear systems using the AMD ordering method. Since the Windows implementation does not use the full SuiteSparse suite, it is however not possible to use the METIS ordering method on Windows. The METIS method can handle larger matrices than the standard AMD method.

Ordering methods

UMFPACK can use several ordering methods. As explained in the previous section, on Windows only one ordering method is possible (AMD), but on other platforms the user can select several other ordering methods.

In R package `umfpackr`, the following ordering methods can be chosen:

- AMD. This is the default. AMD is used for the symmetric strategy and COLAMD for the unsymmetric strategy.
- CHOLMOD. It first tries AMD or COLAMD (depending on what strategy is used). If that method gives low fill-in, it is used without trying METIS at all. Otherwise METIS is tried, and the ordering (AMD/COLAMD or METIS) giving the lowest fill-in is used.
- METIS. Use METIS.

- BEST. Try three methods (AMD/COLAMD, METIS and NESDIS) and take the best. NESDIS is CHOLMOD's nested dissection ordering, based on METIS and CCAMD/CCOLAMD. This results the highest analysis time, but the lowest numerical factorisation time.

The AMD method is usually slightly faster than the other ordering methods, but METIS can handle larger matrices than AMD. Therefore is sometimes necessary to the the METIS ordering (in that case METIS ordering is usually selected when CHOLMOD ordering is used).

More details can be found in the *UMFPACK User Guide* and references therein.

In `umfpackr` the ordering method can be selected by specifying argument `umf_control` of function `umf_solve` or `umf_solve_nl`. This argument should be a named list. To select METIS ordering, specify `umf_control = list(ordering = "METIS")`. An example were the CHOLMOD is chosen on Linux and AMD on Windows:

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

jacdsln <- function(x) {
   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)

# use CHOLMOD ordering on Linux and AMD on Windows
ord <- if (.Platform$OS.type != "windows")  "CHOLMOD" else "AMD"
umf_solve_nl(xstart, dslnex, jacdsln, umf_control = list(ordering = ord))
```

```
Convergence after 7 iterations

$solved
[1] TRUE

$iter
[1] 7

$x
[1] 1 1

$fval
[1] 8.747669e-12 2.215961e-11

$message
[1] "ok"
```