

Package ‘gmresls’

July 22, 2025

Title Solve Least Squares with GMRES(k)

Version 0.2.3

Date 2025-01-17

Description Solves a least squares system $Ax=b$ ($\dim(A)=(m,n)$ with $m \geq n$) with a precondition matrix B : $BAx=Bb$ ($\dim(B)=(n,m)$). Implemented method is based on GMRES (Saad, Youcef; Schultz, Martin H. (1986). "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems" <[doi:10.1137/0907058](https://doi.org/10.1137/0907058)>) with callback functions, i.e. no explicit A , B or b are required.

License GPL (≥ 3)

Encoding UTF-8

RoxygenNote 7.3.2

Suggests RUnit

BugReports <https://forgemia.inra.fr/mathscell/gmresls/-/issues>

NeedsCompilation no

Author Serguei Sokol [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-5674-3327>>)

Maintainer Serguei Sokol <sokol@insa-toulouse.fr>

Repository CRAN

Date/Publication 2025-01-17 15:20:01 UTC

Contents

gmresls	2
Index	4

gmresls

*Solve a Least Squares System with a Preconditioner.***Description**

Solve a least squares system $Ax \approx b$ ($\dim(A)=(m,n)$ with $m \geq n$) with a preconditioner B : $BAx=Bb$ ($\dim(B)=(n,m)$). Implemented method uses GMRES(k) with callback functions, i.e. no explicit A or B are required. GMRES can be restarted after k iterations.

Usage

```
gmresls(f_resid, f_BAx, x0 = NULL, k = 0, maxit = 0, tol = 1e-07, ...)
```

Arguments

<code>f_resid</code>	A function <code>f_resid(x, ...)</code> calculating $B(b-Ax)$ for a given x . If x is of length 0 (e.g. <code>NULL</code>), it must be considered as 0. If x_0 is not <code>NULL</code> in the call to <code>gmres()</code> then x won't be <code>NULL</code> in the <code>f_resid(x, ...)</code> call either.
<code>f_BAx</code>	A function <code>f_BAx(x, ...)</code> calculating matrix-matrix-vector product BAx for a given x .
<code>x0</code>	A vector or <code>NULL</code> (which means 0), initial approximation for $Ax=b$
<code>k</code>	An integer, parameter for restarting GMRES. Value 0 (default) means no restart, i.e. at most $\text{length}(x)$ basis vectors will be constructed and used.
<code>maxit</code>	A maximal iteration number. Here, iteration number continues to increment even after a possible GMRES restart. Default (0) means $\text{length}(x)$.
<code>tol</code>	A tolerance for solution x , estimated as $\ B(Ax-b)\ /\ Bb\ $, default $1.e-7$
<code>...</code>	Parameters passed through to <code>f_BAx</code> and <code>f_resid</code>

Details

Implemented method is equivalent to a classical GMRES(k) method with restart after constructing k basis vectors and applied to a square system $BAx=Bb$. Dense matrices constructed and stored by this method are of size $(\text{length}(x), k)$ and $(k+1, k)$ where k is GMRES current basis vector number. If $\text{maxit} > k$, GMRES will be restarted after each k iterations. Particularity of this implementation that matrices A and B have no to be stored explicitly. User provides just callback function mimicking their multiplication by adequate vectors. In case of non convergence after maxit iterations, `attr(x)` will contain a field 'warning' with the message which will be also issued with `warning()`. If the operator BA is not of full rank, iterations will be stopped before reaching convergence or maxit . A warning will be emitted in this case.

Value

The solution x , having the structure of `f_resid(x,...)`.

Examples

```
# prepare a 4x3 toy least squares (LS) problem Ax=b
A=rbind(diag(1:3)+matrix(1, 3,3), rep(1, 3))
xsol=1:3
b=A%%xsol+rnorm(4, 0., 0.1) # add some noise as it is often the case in LS
f_resid=function(x,...)
  with(list(...), if (length(x) == 0L) crossprod(A, b) else crossprod(A, b-A%%x))
f_BAx=function(x,...)
  with(list(...), crossprod(A, A%%x))
x=gmresls(f_resid, f_BAx, A=A, b=b)
stopifnot(all.equal(c(x), c(qr.solve(A,b))))
```

Index

gmresls, [2](#)