

Package: cPCG (via r-universe)

August 30, 2024

Type Package

Title Efficient and Customized Preconditioned Conjugate Gradient Method for Solving System of Linear Equations

Version 1.0

Date 2018-12-30

Author Yongwen Zhuang

Maintainer Yongwen Zhuang <zyongwen@umich.edu>

Description Solves system of linear equations using (preconditioned) conjugate gradient algorithm, with improved efficiency using Armadillo templated 'C++' linear algebra library, and flexibility for user-specified preconditioning method. Please check <<https://github.com/styvon/cPCG>> for latest updates.

Depends R (>= 3.0.0)

License GPL (>= 2)

Imports Rcpp (>= 0.12.19)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-01-11 17:00:10 UTC

Contents

cPCG-package	2
egsolve	3
icc	4
pcgsolve	5

Index	7
--------------	----------

cPCG-package

*Efficient and Customized Preconditioned Conjugate Gradient Method
for Solving System of Linear Equations*

Description

Solves system of linear equations using (preconditioned) conjugate gradient algorithm, with improved efficiency using Armadillo templated 'C++' linear algebra library, and flexibility for user-specified preconditioning method. Please check <https://github.com/styvon/cPCG> for latest updates.

Details

Functions in this package serve the purpose of solving for x in $Ax = b$, where A is a symmetric and positive definite matrix, b is a column vector.

To improve scalability of conjugate gradient methods for larger matrices, the Armadillo templated C++ linear algebra library is used for the implementation. The package also provides flexibility to have user-specified preconditioner options to cater for different optimization needs.

The DESCRIPTION file:

```
Package:      cPCG
Type:         Package
Title:        Efficient and Customized Preconditioned Conjugate Gradient Method for Solving System of Linear Equations
Version:      1.0
Date:         2018-12-30
Author:       Yongwen Zhuang
Maintainer:   Yongwen Zhuang <zyongwen@umich.edu>
Description:  Solves system of linear equations using (preconditioned) conjugate gradient algorithm, with improved efficiency
Depends:      R (>= 3.0.0)
License:      GPL (>= 2)
Imports:      Rcpp (>= 0.12.19)
LinkingTo:    Rcpp, RcppArmadillo
RoxygenNote: 6.1.1
Encoding:     UTF-8
Suggests:    knitr, rmarkdown
VignetteBuilder: knitr
```

Index of help topics:

```
cPCG-package      Efficient and Customized Preconditioned
                   Conjugate Gradient Method for Solving System of
                   Linear Equations
cgsolve           Conjugate gradient method
icc               Incomplete Cholesky Factorization
pcgsolve         Preconditioned conjugate gradient method
```

Author(s)

Yongwen Zhuang

References

- [1] Reeves Fletcher and Colin M Reeves. “Function minimization by conjugate gradients”. In: The computer journal 7.2 (1964), pp. 149–154.
- [2] David S Kershaw. “The incomplete Cholesky—conjugate gradient method for the iterative solution of systems of linear equations”. In: Journal of computational physics 26.1 (1978), pp. 43–65.
- [3] Yousef Saad. Iterative methods for sparse linear systems. Vol. 82. siam, 2003.
- [4] David Young. “Iterative methods for solving partial difference equations of elliptic type”. In: Transactions of the American Mathematical Society 76.1 (1954), pp. 92–111.

Examples

```
# generate test data
test_A <- matrix(c(4,1,1,3), ncol = 2)
test_b <- matrix(1:2, ncol = 1)

# conjugate gradient method solver
cgsolve(test_A, test_b, 1e-6, 1000)

# preconditioned conjugate gradient method solver,
# with incomplete Cholesky factorization as preconditioner
pcgsolve(test_A, test_b, "ICC")
```

cgsolve

*Conjugate gradient method***Description**

Conjugate gradient method for solving system of linear equations $Ax = b$, where A is symmetric and positive definite, b is a column vector.

Usage

```
cgsolve(A, b, tol = 1e-6, maxIter = 1000)
```

Arguments

<code>A</code>	matrix, symmetric and positive definite.
<code>b</code>	vector, with same dimension as number of rows of A .
<code>tol</code>	numeric, threshold for convergence, default is $1e-6$.
<code>maxIter</code>	numeric, maximum iteration, default is 1000 .

Details

The idea of conjugate gradient method is to find a set of mutually conjugate directions for the unconstrained problem

$$\operatorname{argmin}_x f(x)$$

where $f(x) = 0.5b^T Ax - bx + z$ and z is a constant. The problem is equivalent to solving $Ax = b$.

This function implements an iterative procedure to reduce the number of matrix-vector multiplications [1]. The conjugate gradient method improves memory efficiency and computational complexity, especially when A is relatively sparse.

Value

Returns a vector representing solution x .

Warning

Users need to check that input matrix A is symmetric and positive definite before applying the function.

References

[1] Yousef Saad. Iterative methods for sparse linear systems. Vol. 82. siam, 2003.

See Also

[pcgsolve](#)

Examples

```
## Not run:
test_A <- matrix(c(4,1,1,3), ncol = 2)
test_b <- matrix(1:2, ncol = 1)
cgsolve(test_A, test_b, 1e-6, 1000)

## End(Not run)
```

icc

Incomplete Cholesky Factorization

Description

Incomplete Cholesky factorization method to generate preconditioning matrix for conjugate gradient method.

Usage

```
icc(A)
```

Arguments

A matrix, symmetric and positive definite.

Details

Performs incomplete Cholesky factorization on the input matrix A, the output matrix is used for preconditioning in `pcgsolve()` if "ICC" is specified as the preconditioner.

Value

Returns a matrix after incomplete Cholesky factorization.

Warning

Users need to check that input matrix A is symmetric and positive definite before applying the function.

See Also

[pcgsolve](#)

Examples

```
## Not run:
test_A <- matrix(c(4,1,1,3), ncol = 2)
out <- icc(test_A)

## End(Not run)
```

pcgsolve

Preconditioned conjugate gradient method

Description

Preconditioned conjugate gradient method for solving system of linear equations $Ax = b$, where A is symmetric and positive definite, b is a column vector.

Usage

```
pcgsolve(A, b, preconditioner = "Jacobi", tol = 1e-6, maxIter = 1000)
```

Arguments

A matrix, symmetric and positive definite.
b vector, with same dimension as number of rows of A.
preconditioner string, method for preconditioning: "Jacobi" (default), "SSOR", or "ICC".
tol numeric, threshold for convergence, default is 1e-6.
maxIter numeric, maximum iteration, default is 1000.

Details

When the condition number for A is large, the conjugate gradient (CG) method may fail to converge in a reasonable number of iterations. The Preconditioned Conjugate Gradient (PCG) Method applies a precondition matrix C and approaches the problem by solving:

$$C^{-1}Ax = C^{-1}b$$

where the symmetric and positive-definite matrix C approximates A and $C^{-1}A$ improves the condition number of A .

Common choices for the preconditioner include: Jacobi preconditioning, symmetric successive over-relaxation (SSOR), and incomplete Cholesky factorization [2].

Value

Returns a vector representing solution x .

Preconditioners

Jacobi: The Jacobi preconditioner is the diagonal of the matrix A , with an assumption that all diagonal elements are non-zero.

SSOR: The symmetric successive over-relaxation preconditioner, implemented as $M = (D+L)D^{-1}(D+L)^T$. [1]

ICC: The incomplete Cholesky factorization preconditioner. [2]

Warning

Users need to check that input matrix A is symmetric and positive definite before applying the function.

References

[1] David Young. “Iterative methods for solving partial difference equations of elliptic type”. In: Transactions of the American Mathematical Society 76.1 (1954), pp. 92–111.

[2] David S Kershaw. “The incomplete Cholesky—conjugate gradient method for the iterative solution of systems of linear equations”. In: Journal of computational physics 26.1 (1978), pp. 43–65.

See Also

[cgsolve](#)

Examples

```
## Not run:
test_A <- matrix(c(4,1,1,3), ncol = 2)
test_b <- matrix(1:2, ncol = 1)
pcgsolve(test_A, test_b, "ICC")

## End(Not run)
```

Index

* **methods**

 cgsolve, 3

 icc, 4

 pcgsolve, 5

* **optimize**

 cgsolve, 3

 pcgsolve, 5

* **package**

 cPCG-package, 2

cgsolve, 3, 6

cPCG (cPCG-package), 2

cPCG-package, 2

icc, 4

pcgsolve, 4, 5, 5

preconditioner (pcgsolve), 5