Package 'RcppAMGCL'

November 26, 2026

Type Package

Title Sparse Iterative Solvers with Algebraic Multigrid Preconditioning

Version 0.1.0

Description Solve sparse linear systems using conjugate gradients or stabilized bi-conjugate gradients with algebraic multigrid preconditioning from the header-only 'AMGCL' library described by Demidov (2020) <doi:10.1016/j.simpa.2020.100037>. Supports compressed sparse column matrices, reusable solver objects, and optional 'OpenMP' acceleration with explicit control over the number of threads. Provides a reusable C++ interface for downstream packages.

License MIT + file LICENSE

Copyright See inst/COPYRIGHTS for all bundled-code copyright holders and licenses, including institutional and project contributors.

URL https://github.com/santiagohermo/RcppAMGCL

BugReports https://github.com/santiagohermo/RcppAMGCL/issues

Encoding UTF-8

Depends R (>= 4.3.0)

Imports Matrix, methods, Rcpp

LinkingTo Rcpp

SystemRequirements GNU make; OpenMP (optional)

NeedsCompilation yes

Author Santiago Hermo [aut, cre, cph],
Denis Demidov [ctb, cph] (Author of the bundled AMGCL library),
Riccardo Rossi [ctb, cph] (Bundled AMGCL and Kratos-derived code),
Pooyan Dadvand [ctb, cph] (Bundled Kratos-derived code),
Mohammad Siahatgar [ctb, cph] (Bundled AMGCL performance counters),
Christoph Sohrmann [ctb, cph] (Bundled AMGCL complex value types),
Peter Gamnitzer [ctb, cph] (Bundled AMGCL Chebyshev relaxation)

Maintainer Santiago Hermo <santiago.hermo@monash.edu>

Repository CRAN

Date/Publication 2026-11-26 11:10:10 UTC

Contents

RcppAMGCL-package 2
amgcl_info 2
amgcl_setup 3
amgcl_solve 5

Index **7**

`RcppAMGCL-package` *Sparse Iterative Solvers with Algebraic Multigrid Preconditioning*

Description

An R interface to the header-only AMGCL library using Rcpp, with a built-in CPU backend and optional OpenMP acceleration. Use `amgcl_solve` for a single sparse system, or `amgcl_setup` to reuse a preconditioner across right-hand sides.

Details

The package supports general and symmetric double-precision compressed sparse column matrices (`dgCMatrix` and `dsCMatrix`) from **Matrix**, with vector or one-column matrix right-hand sides. See `amgcl_info` for build capabilities. The installed '`RcppAMGCL.h`' header provides an R-independent C++ interface for downstream code; the bundled AMGCL headers retain their upstream license notices. This product includes Kratos Multi-Physics technology. The installed '`COPYRIGHTS`' file inventories bundled-code ownership and licenses; additional notices are installed under '`licenses`'. Use `citation("RcppAMGCL")` for references to this package and AMGCL.

References

Demidov, D. (2020). AMGCL – A C++ library for efficient solution of large sparse linear systems. *Software Impacts*, 6, 100037. doi:10.1016/j.simpa.2020.100037.

`amgcl_info` *Report AMGCL and OpenMP Capabilities*

Description

Report the bundled AMGCL version and OpenMP capabilities of the installed package. Querying this information does not change OpenMP settings.

Usage

```
amgcl_info()
```

Details

Parallelism is opt-in: `amgcl_setup()` and `amgcl_solve()` default to one thread even if OpenMP is enabled. Set `OMP_THREAD_LIMIT` before starting R to impose a process-wide OpenMP thread limit. Runtime resource constraints can result in fewer worker threads than requested.

Value

A list with components:

openmp Logical; whether this package was compiled with OpenMP.

max_threads Available thread capacity, capped by processor count and the OpenMP thread limit, and at least one.

thread_limit OpenMP thread limit, reflecting `OMP_THREAD_LIMIT`. One for a serial build.

current_threads Current calling-task OpenMP thread-count setting. One for a serial build.

amgcl_version Version of the bundled AMGCL library.

See Also

`amgcl_setup`, `amgcl_solve`

Examples

```
amgcl_info()
```

---

amgcl_setup *Create a Reusable Algebraic Multigrid Solver*

Description

Build an algebraic multigrid (AMG) preconditioner and retain it for repeated solutions with the same matrix. The matrix is copied into storage owned by the solver, so subsequent changes to the R matrix do not change the solver.

Usage

```
amgcl_setup(A, solver = c("cg", "bicgstab"), tol = 1e-8,
            maxiter = 1000L, threads = 1L)

## S3 method for class 'amgcl_solver'
print(x, ...)
```

Arguments

A	A nonempty, square Matrix::dgCMatrix or symmetric Matrix::dsCMatrix containing finite double-precision values and a nonzero diagonal entry in every row. Symmetric storage is expanded internally after validation.
solver	Iterative method: "cg" for conjugate gradients or "bicgstab" for stabilized bi-conjugate gradients.
tol	Finite positive convergence tolerance for AMGCL's relative residual estimate.
maxiter	Positive integer maximum number of iterations.
threads	Positive integer requested thread count, defaulting to one. Counts above the available capacity are capped. A build without OpenMP always uses one thread.
x	An amgcl_solver object.
...	Additional arguments, currently unused.

Details

The package uses AMGCL's built-in CPU backend, smoothed aggregation and SPAI0 relaxation. A nonzero diagonal is required by the selected AMG hierarchy and its unpivoted coarse solver; matrices with a missing or zero diagonal entry are rejected. Conjugate gradients require a symmetric positive definite matrix. The package does not verify positive definiteness, and convergence does not establish that this requirement is satisfied. BiCGStab accepts nonsymmetric systems. Convergence is not guaranteed; singular systems and nullspace constraints are not handled automatically.

Systems with at most 64 rows use the direct coarse LU solver as their preconditioner and often converge in one iteration. Hierarchy setup errors include the setup context and underlying AMGCL diagnostic; they do not by themselves establish that the matrix is singular.

The solver owns its method, tolerance and iteration limit. These cannot be overridden in subsequent calls to amgcl_solve(); create a new solver to change them. The threads argument applies only to the current operation: set it separately when solving with this object.

OpenMP, when available, accelerates both setup and solving. Thread counts are capped by processor capacity and OMP_THREAD_LIMIT. OpenMP may provide fewer worker threads than requested. The calling task's thread count and dynamic-thread setting are restored after each operation. No R API functions are called from worker threads.

Solver objects contain an R external pointer and are valid only in the process where they were created. Do not save and restore them with saveRDS(), send them to another R process, or access the same object concurrently. Recreate a solver from its matrix in each process. Native storage is released automatically when the object is garbage collected.

Value

An object of class amgcl_solver, containing an external pointer and metadata n, solver, tol and maxiter. Treat its components as read-only. The print method returns the object invisibly.

See Also

amgcl_solve, amgcl_info

Examples

```
A <- Matrix::bandSparse(10, k = c(0, 1),
    diagonals = list(rep(3, 10), rep(-1, 9)), symmetric = TRUE)
s <- amgcl_setup(A)
print(s)
amgcl_solve(s, rep(1, 10))$x
amgcl_solve(s, seq_len(10))$residual
```

`amgcl_solve` *Solve a Sparse Linear System*

Description

Solve $Ax = b$ with AMG-preconditioned conjugate gradients or BiCGStab, either by constructing a solver for one call or reusing an existing solver.

Usage

```
amgcl_solve(A, b, solver = c("cg", "bicgstab"), tol = 1e-8,
            maxiter = 1000L, threads = 1L)
```

Arguments

`A`	A nonempty, square `Matrix::dgCMatrix` or symmetric `Matrix::dsCMatrix` with finite entries and a nonzero diagonal entry in every row, or a reusable `amgcl_solver` created by `amgcl_setup`.
`b`	A finite real numeric right-hand-side vector, including an integer vector, of length equal to the matrix dimension. A one-column base numeric matrix or one-column double-precision **Matrix** object with the same number of rows is also accepted and converted to a vector. Multiple columns, arrays with more than two dimensions, and logical or complex inputs are rejected.
`solver`	`"cg"` or `"bicgstab"`. Must be omitted when `A` is a reusable solver.
`tol`	Finite positive tolerance for AMGCL's relative residual estimate. Must be omitted when `A` is a reusable solver.
`maxiter`	Positive integer iteration limit. Must be omitted when `A` is a reusable solver.
`threads`	Positive integer requested thread count for this call, defaulting to one. Capped by the capacity reported by `amgcl_info`; one thread is used without OpenMP.

Details

Every solve starts from a zero initial guess. Inspect `converged` and `residual` before using a result. CG requires a symmetric positive definite matrix; convergence does not validate that assumption. See `amgcl_setup` for matrix requirements, the direct coarse LU preconditioner used for systems with at most 64 rows, and setup errors.

When `A` is a matrix, `threads` controls both setup and solving. When it is a reusable solver, `threads` controls this solve only; the setup thread count is not retained. OpenMP settings are restored after each operation. See `amgcl_setup` for thread and object-lifetime details.

Value

A list with components:

x Numeric solution vector.

iterations Number of iterations reported by AMGCL.

error AMGCL's final relative residual estimate.

residual Independently computed Euclidean residual norm divided by the Euclidean norm of `b`. For a zero right-hand side, the absolute residual norm is returned.

converged Logical; whether the independently computed `residual` is at most `tol`.

threads Effective requested thread cap after applying build and capacity limits. This is not a measurement of the worker count used inside each parallel region.

See Also

`amgcl_setup`, `amgcl_info`

Examples

```
A <- methods::as(Matrix::bandSparse(10, k = c(-1, 0, 1),
    diagonals = list(rep(-1, 9), rep(3, 10), rep(-1, 9))), "dgCMatrix")
result <- amgcl_solve(A, rep(1, 10))
result$x
result$residual

# Symmetric sparse storage and a one-column Matrix right-hand side.
S <- Matrix::bandSparse(10, k = c(0, 1),
    diagonals = list(rep(3, 10), rep(-1, 9)), symmetric = TRUE)
rhs <- S %*% rep(1, 10)
amgcl_solve(S, rhs)$x
```

Index

amgcl_info, *2*, 2, *4–6*
amgcl_setup, *2, 3*, 3, *5, 6*
amgcl_solve, *2–4*, 5
amgcl_solver *(*amgcl_setup*)*, 3

print.amgcl_solver *(*amgcl_setup*)*, 3

RcppAMGCL *(*RcppAMGCL-package*)*, 2
RcppAMGCL-package, 2