Package 'ECOSolveR'

February 18, 2026

Type Package

Title Embedded Conic Solver in R

Version 0.6.1

Date 2026-02-18

VignetteBuilder knitr

SystemRequirements GNU make

URL https://bnaras.github.io/ECOSolveR/

BugReports https://github.com/bnaras/ECOSolveR/issues

Imports cli, methods

Suggests knitr, rmarkdown, testthat, Matrix, covr, slam

Description R interface to the Embedded COnic Solver (ECOS), an efficient and robust C library for convex problems. Conic and equality constraints can be specified in addition to integer and boolean variable constraints for mixed-integer problems. This R interface is inspired by the python interface and has similar calling conventions.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author Anqi Fu [aut],
Balasubramanian Narasimhan [aut, cre],
Florian Schwendinger [ctb],
Martin Maechler [ctb]

Maintainer Balasubramanian Narasimhan <naras@stat.Stanford.EDU>

Repository CRAN

Date/Publication 2026-02-18 15:30:02 UTC

Contents

`ecos.control` *Return the default optimization parameters for ECOS*

Description

This is used to control the behavior of the underlying optimization code.

Usage

```
ecos.control(
  maxit = 100L,
  feastol = 1e-08,
  reltol = 1e-08,
  abstol = 1e-08,
  feastol_inacc = 1e-04,
  abstol_inacc = 5e-05,
  reltol_inacc = 5e-05,
  verbose = 0L,
  mi_max_iters = 1000L,
  mi_int_tol = 1e-04,
  mi_abs_eps = 1e-06,
  mi_rel_eps = 1e-06
)
```

Arguments

`maxit`	the maximum number of iterations for ecos, default 100L
`feastol`	the tolerance on the primal and dual residual, default 1e-8
`reltol`	the relative tolerance on the duality gap, default 1e-8
`abstol`	the absolute tolerance on the duality gap, default 1e-8
`feastol_inacc`	the tolerance on the primal and dual residual if reduced precisions, default 1e-4
`abstol_inacc`	the absolute tolerance on the duality gap if reduced precision, default 5e-5
`reltol_inacc`	the relative tolerance on the duality gap if reduced precision, default 5e-5

verbose	verbosity level, default 0L. A verbosity level of 1L will show more detail, but clutter session transcript.
mi_max_iters	the maximum number of branch and bound iterations (mixed integer problems only), default 1000L
mi_int_tol	the integer tolerence (mixed integer problems only), default 1e-4
mi_abs_eps	the absolute tolerance between upper and lower bounds (mixed integer problems only), default 1e-6
mi_rel_eps	the relative tolerance, $(U - L)/L$, between upper and lower bounds (mixed integer problems only), default 1e-6

Value

a list with the following elements:

FEASTOL the tolerance on the primal and dual residual, parameter `feastol`

ABSTOL the absolute tolerance on the duality gap, parameter `abstol`

RELTOL the relative tolerance on the duality gap, parameter `reltol`

FEASTOL_INACC the tolerance on the primal and dual residual if reduced precisions, parameter `feastol_inacc`

ABSTOL_INACC the absolute tolerance on the duality gap if reduced precision, parameter `abstol_inacc`

RELTOL_INACC the relative tolerance on the duality gap if reduced precision, parameter `reltol_inacc`

MAXIT the maximum number of iterations for ecos, parameter `maxit`

MI_MAX_ITERS the maximum number of branch and bound iterations (mixed integer problems only), parameter `mi_max_iters`

MI_INT_TOL the integer tolerence (mixed integer problems only), parameter `mi_int_tol`

MI_ABS_EPS the absolute tolerance between upper and lower bounds (mixed integer problems only), parameter `mi_abs_eps`

MI_REL_EPS the relative tolerance, $(U - L)/L$, between upper and lower bounds (mixed integer problems only), parameter `mi_rel_eps`

VERBOSE verbosity level, parameter `verbose`

`ECOS_cleanup` *Clean up an ECOS workspace*

Description

Frees the C-level ECOS workspace. After this call the workspace external pointer is invalidated and cannot be used for further solves or updates. It is safe to call this more than once.

Usage

```
ECOS_cleanup(workspace)
```

Arguments

workspace	an external pointer of class "ecos_workspace" as returned by ECOS_setup.

Value

NULL, invisibly.

See Also

ECOS_setup, ECOS_solve, ECOS_update

ECOS_csolve *Solve a conic optimization problem*

Description

The function ECOS_csolve is a wrapper around the ecos csolve C function. Conic constraints are specified using the G and h parameters and can be NULL and zero length vector respectively indicating an absence of conic constraints. Similarly, equality constraints are specified via A and b parameters with NULL and empty vector values representing a lack of such constraints. At most one of the pair (G, h) or (A, b) is allowed to be absent.

Usage

```
ECOS_csolve(
  c = numeric(0),
  G = NULL,
  h = numeric(0),
  dims = list(l = integer(0), q = NULL, e = integer(0)),
  A = NULL,
  b = numeric(0),
  bool_vars = integer(0),
  int_vars = integer(0),
  control = ecos.control()
)
```

Arguments

c	the coefficients of the objective function; the length of this determines the number of variables n in the problem.
G	the inequality constraint matrix in one of three forms: a plain matrix, simple triplet matrix, or compressed column format, e.g. dgCMatrix-class. Can also be NULL
h	the right hand side of the inequality constraint. Can be empty numeric vector.

`dims`	is a list of three named elements: `dims['l']` an integer specifying the dimension of positive orthant cone, `dims['q']` an integer vector specifying dimensions of second-order cones, `dims['e']` an integer specifying the number of exponential cones
`A`	the optional equality constraint matrix in one of three forms: a plain matrix, simple triplet matrix, or compressed column format, e.g. dgCMatrix-class. Can be `NULL`
`b`	the right hand side of the equality constraint, must be specified if A is. Can be empty numeric vector.
`bool_vars`	the indices of the variables, 1 through n, that are boolean; that is, they are either present or absent in the solution
`int_vars`	the indices of the variables, 1 through n, that are integers
`control`	is a named list that controls various optimization parameters; see ecos.control.

Value

a list of 8 named items

x primal variables

y dual variables for equality constraints

s slacks for $Gx + s <= h$, $s \in K$

z dual variables for inequality constraints $s \in K$

infostring gives information about the status of solution

retcodes a named integer vector containing four elements

- **exitflag** 0=`ECOS_OPTIMAL`, 1=`ECOS_PINF`, 2=`ECOS_DINF`, 10=`ECOS_INACC_OFFSET`, -1=`ECOS_MAXIT`, -2=`ECOS_NUMERICS`, -3=`ECOS_OUTCONE`, -4=`ECOS_SIGINT`, -7=`ECOS_FATAL`. See ECOS_exitcodes.
- **iter** the number of iterations used
- **mi_iter** the number of iterations for mixed integer problems
- **numerr** a non-zero number if a numeric error occurred

summary a named numeric vector containing

- **pcost** value of primal objective
- **dcost** value of dual objective
- **pres** primal residual on inequalities and equalities
- **dres** dual residual
- **pinf** primal infeasibility measure
- **dinf** dual infeasibility measure
- **pinfres** primal infeasibility residual
- **dinfres** dual infeasibility residual
- **gap** duality gap
- **relgap** relative duality gap
- **r0** Unknown at the moment to this R package maintainer.

timing a named numeric vector of timing information consisting of

- **runtime** the total runtime in ecos
- **tsetup** the time for setup of the problem
- **tsolve** the time to solve the problem

Details

A call to this function will solve the problem: minimize $c^T x$, subject to $Ax = b$, and $h - G * x \in K$.

Variables can be constrained to be boolean (1 or 0) or integers. This is indicated by specifying parameters `bool_vars` and/or `int_vars` respectively. If so indicated, the solutions will be found using a branch and bound algorithm.

Examples

```
## githubIssue98
cat("Basic matrix interface\n")
Gmat <- matrix(c(0.416757847405471, 2.13619609566845, 1.79343558519486, 0, 0,
              0, 0, -1, 0, 0, 0, 0.056266827226329, -1.64027080840499, 0.841747365656204,
                 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0.416757847405471, 2.13619609566845,
           1.79343558519486, 0, 0, 0, -1, 0, 0, 0, 0, 0.056266827226329, -1.64027080840499,
            0.841747365656204, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0), ncol = 5L)
c <- as.numeric(c(0, 0, 0, 0, 1))
h <- as.numeric(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
dims <- list(l = 6L, q = 5L, e = 0L)
ECOS_csolve(c = c, G = Gmat, h = h,
           dims = dims,
           A = NULL, b = numeric(0))

cat("Simple Triplet Matrix interface, if you have package slam\n")
if (requireNamespace("slam")) {

  ECOS_csolve(c = c, G = slam::as.simple_triplet_matrix(Gmat), h = h,
              dims = dims,
              A = NULL, b = numeric(0))
}

if (requireNamespace("Matrix")) {
   ECOS_csolve(c = c, G = Matrix::Matrix(Gmat), h = h,
               dims = dims,
               A = NULL, b = numeric(0))
}

## Larger problems using saved data can be found in the test suite.
## Here is one
if (requireNamespace("Matrix")) {
  MPC01 <- readRDS(system.file("testdata", "MPC01_1.RDS", package = "ECOSolveR"))
  G <- Matrix::sparseMatrix(x = MPC01$Gpr, i = MPC01$Gir, p = MPC01$Gjc,
                            dims = c(MPC01$m, MPC01$n), index1 = FALSE)
  h <- MPC01$h
  dims <- lapply(list(l = MPC01$l, q=MPC01$q, e=MPC01$e), as.integer)
  retval <- ECOS_csolve(c = MPC01$c, G=G, h = h, dims = dims, A = NULL, b = NULL,
                        control = ecos.control(verbose=1L))
  retval$retcodes
  retval$infostring
  retval$summary
}
```

ECOS_exitcodes *ECOS solver exit codes*

Description

A two-column data frame consisting of the code and description for the ECOS solver with ECOS symbolic code names as row names

ECOS_setup *Set up an ECOS workspace for multi-step solving*

Description

Creates an ECOS workspace that can be solved, updated with new numerical data, and solved again without repeating the expensive symbolic analysis phase. This is useful for parametric optimization, model predictive control, and other settings where the problem structure stays the same but data changes.

Usage

```
ECOS_setup(
  c,
  G,
  h,
  dims = list(l = integer(0), q = NULL, e = integer(0)),
  A = NULL,
  b = numeric(0),
  control = ecos.control()
)
```

Arguments

c	the coefficients of the objective function; the length of this determines the number of variables n in the problem.
G	the inequality constraint matrix in one of three forms: a plain matrix, simple triplet matrix, or compressed column format, e.g. dgCMatrix-class. Can also be NULL
h	the right hand side of the inequality constraint. Can be empty numeric vector.
dims	is a list of three named elements: dims['l'] an integer specifying the dimension of positive orthant cone, dims['q'] an integer vector specifying dimensions of second-order cones, dims['e'] an integer specifying the number of exponential cones
A	the optional equality constraint matrix in one of three forms: a plain matrix, simple triplet matrix, or compressed column format, e.g. dgCMatrix-class. Can be NULL

b	the right hand side of the equality constraint, must be specified if A is. Can be empty numeric vector.
control	is a named list that controls various optimization parameters; see ecos.control.

Value

an external pointer of class "ecos_workspace". Must eventually be freed via ECOS_cleanup or R garbage collection.

See Also

ECOS_solve, ECOS_update, ECOS_cleanup, ECOS_csolve

ECOS_solve *Solve an ECOS workspace*

Description

Calls ECOS_solve on an existing workspace created by ECOS_setup. The workspace can be solved multiple times, optionally with ECOS_update calls in between to change numerical data.

Usage

```
ECOS_solve(workspace, control = NULL)
```

Arguments

workspace	an external pointer of class "ecos_workspace" as returned by ECOS_setup.
control	optional named list of solver parameters (see ecos.control). If NULL (the default), the settings from ECOS_setup (or the most recent ECOS_solve call with non-NULL control) are reused.

Value

the same result list as ECOS_csolve.

See Also

ECOS_setup, ECOS_update, ECOS_cleanup

ECOS_update *Update numerical data in an ECOS workspace*

Description

Replaces numerical values in an existing ECOS workspace without repeating symbolic analysis. The sparsity structure must remain the same; only the non-zero values of G and A, and the dense vectors c, h, b can be updated. Pass `NULL` for any argument to leave it unchanged.

Usage

```
ECOS_update(workspace, Gpr = NULL, Apr = NULL, c = NULL, h = NULL, b = NULL)
```

Arguments

`workspace`	an external pointer of class `"ecos_workspace"` as returned by `ECOS_setup`.
`Gpr`	numeric vector of new non-zero values for G (same length as original), or `NULL` to keep current values.
`Apr`	numeric vector of new non-zero values for A (same length as original), or `NULL` to keep current values.
`c`	numeric vector of new objective coefficients, or `NULL`.
`h`	numeric vector of new inequality RHS, or `NULL`.
`b`	numeric vector of new equality RHS, or `NULL`.

Value

the `workspace` object, invisibly.

See Also

`ECOS_setup`, `ECOS_solve`, `ECOS_cleanup`

print.ecos_workspace *Print method for ECOS workspace objects*

Description

Print method for ECOS workspace objects

Usage

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

Arguments

x	an "ecos_workspace" external pointer.
...	ignored.

Value

x, invisibly.

Index