Package 'Uno'

June 8, 2026

Type Package

Title R Interface to the 'Uno' Nonlinear Optimization Solver

Version 2.7.3

Description Bindings to 'Uno' (Unifying Nonlinear Optimization), a C++ solver for smooth nonlinearly constrained optimization. 'Uno' unifies Lagrange-Newton methods, including sequential quadratic programming and interior-point methods, by decomposing them into interacting building blocks (constraint-relaxation, inequality-handling, Hessian, and globalization strategies) that can be freely combined, either through options or through presets that reproduce established solvers such as 'filterSQP' and 'IPOPT'. The framework is described in Vanaret and Leyffer (2024) <doi:10.48550/arXiv.2406.13454>.

License MIT + file LICENSE

Copyright file inst/COPYRIGHTS

URL https://bnaras.github.io/Uno/, https://github.com/bnaras/Uno

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

SystemRequirements CMake (>= 3.16), C++17, GNU make

Imports rmumps (>= 5.2.1-41)

LinkingTo cpp11, rmumps

Suggests cpp11, tinytest, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 7.3.3

Encoding UTF-8

Author Balasubramanian Narasimhan [aut, cre],
Charlie Vanaret [aut, cph] (Designed and implemented Uno),
Sven Leyffer [aut, cph] (Co-developed the Uno framework),
HiGHS development team [cph] (Bundled HiGHS LP/QP/MIP solver (MIT); see inst/COPYRIGHTS)

Maintainer Balasubramanian Narasimhan <naras@stanford.edu>

Repository CRAN

Date/Publication 2026-06-08 19:20:02 UTC

Contents

`uno_solve` *Solve a nonlinear program with Uno*

Description

Solve a nonlinear program with Uno

Usage

```
uno_solve(
  n,
  lb,
  ub,
  sense,
  obj,
  grad,
  m,
  cl,
  cu,
  cons,
  jac_rows,
  jac_cols,
  jac,
  hess_rows,
  hess_cols,
  hess,
  x0,
  preset,
  base_indexing,
  verbose,
  options = list(),
  lagrangian_sign = c("negative", "positive"),
  dual0 = NULL,
  iter_callback = NULL,
  log_callback = NULL
)
```

Arguments

n	number of variables.
lb, ub	variable lower/upper bounds (length ‘n‘; use ‘-Inf‘/‘Inf‘).
sense	‘"minimize"‘ or ‘"maximize"‘.
obj, grad	objective ‘function(x)‘ and its gradient ‘function(x)‘.
m	number of constraints (0 for unconstrained).
cl, cu	constraint lower/upper bounds (length ‘m‘).
cons	constraint ‘function(x)‘ returning a length-‘m‘ vector.
jac_rows, jac_cols	COO row/column indices of the Jacobian nonzeros.
jac	Jacobian ‘function(x)‘ returning the nonzero values.
hess_rows, hess_cols	COO indices of the lower-triangular Hessian.
hess	Lagrangian Hessian ‘function(x, sigma, lambda)‘ returning the lower-triangular nonzero values, or ‘NULL‘ (Uno then uses an L-BFGS approximation, which the HiGHS subproblem solver cannot use).
x0	initial primal iterate (length ‘n‘).
preset	Uno preset, e.g. ‘"filtersqp"‘ (SQP) or ‘"ipopt"‘ (interior point, using MUMPS as the linear solver).
base_indexing	0 for C-style or 1 for Fortran-style COO indices.
verbose	if ‘FALSE‘, suppress Uno’s solution printout.
options	a named list of Uno solver options applied AFTER the preset (so they override it), e.g. ‘list(max_iterations = 200L, tolerance = 1e-8, linear_solver = "MUMPS")‘. Each value is coerced to the option’s declared Uno type; an unknown option name or an unacceptable value raises an error.
lagrangian_sign	the Lagrangian multiplier sign convention the ‘hess‘ callback uses: ‘"positive"‘ for $L = \sigma f + y^\top c$ (the standard convention used by IPOPT and the sparsediff oracle) or ‘"negative"‘ for $L = \sigma f - y^\top c$. Defaults to ‘"negative"‘, matching Uno’s own C-API default. **Must match the convention your ‘hess‘ returns**, otherwise the Lagrangian Hessian’s constraint terms get the wrong sign (invisible when all constraints are linear, since their Hessian is zero).
dual0	optional warm-start dual iterate, or ‘NULL‘ (Uno’s default).
iter_callback	optional ‘function(info)‘ called at each acceptable iterate; return ‘TRUE‘ to terminate the solve early. ‘info‘ is a named list with ‘primals‘, ‘lower_bound_dual‘, ‘upper_bound_dual‘, ‘constraint_dual‘, ‘objective_multiplier‘, and the ‘primal_feasibility‘/‘stationarity‘/‘complementarity‘ residuals. Errors in the callback are caught and treated as "do not terminate". ‘NULL‘ disables it.
log_callback	optional ‘function(text)‘ that receives Uno’s output stream in chunks (a sink for the solver log); ‘NULL‘ leaves output on stdout. Independent of ‘verbose‘ (which controls how much Uno prints).

Value

a named list. The ‘optimization_status‘ and ‘solution_status‘ are **named integers** of the form ‘c(SUCCESS = 0L)‘: the value is Uno’s enum code and the name is its canonical label, so you can key a status map by ‘names(status)‘ and still read the code (e.g. ‘status[[1L]]‘). The list also holds the objective, primal and dual solutions (‘constraint_dual‘, ‘lower_bound_dual‘, ‘upper_bound_dual‘), KKT residuals, and per-callback evaluation counters.

`uno_version` *Linked Uno version*

Description

Returns the version string of the Uno C++ library this package is built against.

Value

A character scalar, e.g. `"2.7.2"`.

Examples

```
uno_version()
```

Index