

Package ‘arcopt’

May 7, 2026

Type Package

Title Adaptive Regularization using Cubics for Optimization

Version 0.3.0

Description Implements cubic regularization methods (ARC) for local optimization problems common in statistics and applied research. Provides robust handling of ill-conditioned, nonconvex, and indefinite Hessian problems with automatic saddle point escape. Supports box constraints; linear equality constraints are planned for a future release.

License MIT + file LICENSE

Encoding UTF-8

URL <https://github.com/marcus-waldman/arcopt>

BugReports <https://github.com/marcus-waldman/arcopt/issues>

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.0), utils

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), knitr, rmarkdown, covr, marqLevAlg, trust

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation yes

Author Marcus Waldman [aut, cre]

Maintainer Marcus Waldman <marcus.waldman@cuanschultz.edu>

Repository CRAN

Date/Publication 2026-04-29 08:10:10 UTC

Contents

arcopt	2
arcopt_advanced_controls	4

Index	8
--------------	----------

arcopt

Adaptive Regularization using Cubics Optimizer

Description

Minimizes a nonlinear objective function using Adaptive Regularization with Cubics (ARC). Designed for robust optimization of ill-conditioned, nonconvex, and indefinite Hessian problems common in statistical applications.

Usage

```
arcopt(
  x0,
  fn,
  gr,
  hess = NULL,
  ...,
  lower = rep(-Inf, length(x0)),
  upper = rep(Inf, length(x0)),
  control = list()
)
```

Arguments

<code>x0</code>	Numeric vector of initial parameter values (length Q).
<code>fn</code>	Function that computes the objective function value. Should take a numeric vector of length Q and return a scalar.
<code>gr</code>	Function that computes the gradient. Should take a numeric vector of length Q and return a numeric vector of length Q . Required.
<code>hess</code>	Function that computes the Hessian matrix. Should take a numeric vector of length Q and return a Q -by- Q symmetric matrix. Required (unless <code>control\$use_qn = TRUE</code> ; see Details).
<code>...</code>	Additional arguments passed to <code>fn</code> , <code>gr</code> , and <code>hess</code> .
<code>lower</code>	Numeric vector of lower bounds (length Q). Use <code>-Inf</code> for unbounded parameters. Default: all <code>-Inf</code> .
<code>upper</code>	Numeric vector of upper bounds (length Q). Use <code>Inf</code> for unbounded parameters. Default: all <code>Inf</code> .
<code>control</code>	A named list of control parameters. The user-facing tolerances and switches are documented below; advanced regularization tuning, the trust-region fallback, and the quasi-Newton polish mode live on a separate help page (see \link{arcopt_advanced_controls}). Recognized entries: <code>maxit</code> Maximum number of iterations (default 1000). <code>gtol_abs</code> Absolute gradient-norm tolerance for convergence (default 1e-5). <code>ftol_abs</code> Absolute objective-value tolerance (default 1e-8).

`xtol_abs` Absolute step-size tolerance (default $1e-8$).
`trace` Integer in $0:3$. Depth of per-iteration data captured into `result$trace`:
 0 collects nothing, 1 (the default) collects function value and gradient norm,
 2 adds sigma, rho, step type and reciprocal Hessian condition number, 3
adds the full iterate, step, Hessian and convergence-criterion record. This
flag controls *saved* data only – for live console output see `verbose`.
`verbose` Logical. If TRUE, prints one line per iteration to the console show-
ing iteration number, objective value, $\|g\|_{\infty}$, ratio rho, regularization
scale, and active solver mode (default FALSE). Orthogonal to `trace`.
`use_qn` Logical. If TRUE, route to the quasi-Newton ARC variant, which ap-
proximates the Hessian via SR1/BFGS updates and does not require an
analytic hess function. See the advanced-controls page for QN-specific
parameters (default FALSE).
See [\link{arcopt_advanced_controls}](#) for the full set of advanced tuning
parameters governing the cubic regularization, the trust-region fallback, and the
quasi-Newton polish mode.

Details

The ARC algorithm iteratively minimizes a cubic regularization model:

$$m_k(s) = f_k + g_k^\top s + \frac{1}{2} s^\top H_k s + \frac{\sigma_k}{3} \|s\|^3$$

where σ_k is adapted from observed model accuracy. `arcopt` may transparently fall back to a trust-region subproblem in flat-ridge regimes and (optionally, opt-in) to a line-search BFGS polish in the quadratic attraction basin. The transitions are observable via `result$diagnostics`; the algorithmic details and tunable thresholds are documented under [\link{arcopt_advanced_controls}](#).

Hessian Requirement:

`arcopt` is Hessian-centric: an analytic hess function is strongly recommended. If the analytic form is unavailable, set `control$use_qn = TRUE` to obtain Hessian-free quasi-Newton updates (see the advanced-controls page).

Value

A list with components:

- `par`: Optimal parameter vector.
- `value`: Objective value at `par`.
- `gradient`: Gradient at `par`.
- `hessian`: Hessian at `par` (or the final BFGS approximation if the run ended in `qn_polish` mode).
- `sigma`: Final cubic regularization parameter.
- `converged`: Logical; whether convergence criteria were met.
- `iterations`: Number of iterations performed.
- `evaluations`: Named list of `fn`, `gr`, and `hess` evaluation counts.

- `message`: Convergence reason.
- `trace`: Per-iteration trace data (depth controlled by `control$trace`); NULL when `trace = 0`.
- `diagnostics`: Sublist of internal mode-dispatch diagnostics – `solver_mode_final`, `ridge_switches`, `radius_final`, `qn_polish_switches`, `qn_polish_reverts`, and `hess_evals_at_polish_switch`. See [\link{archopt_advanced_controls}](#) for the meaning of each field. Most users do not need to inspect this; it is preserved for diagnostic and benchmarking use.

See Also

[archopt_advanced_controls](#) for advanced tuning of the cubic regularization, trust-region fallback, and quasi-Newton polish mode.

Examples

```
# Rosenbrock function
rosenbrock <- function(x) (1 - x[1])^2 + 100 * (x[2] - x[1]^2)^2

rosenbrock_gr <- function(x) {
  c(-2 * (1 - x[1]) - 400 * x[1] * (x[2] - x[1]^2),
    200 * (x[2] - x[1]^2))
}

rosenbrock_hess <- function(x) {
  matrix(c(
    1200 * x[1]^2 - 400 * x[2] + 2, -400 * x[1],
    -400 * x[1], 200
  ), 2, 2)
}

result <- archopt(
  x0 = c(-1.2, 1),
  fn = rosenbrock,
  gr = rosenbrock_gr,
  hess = rosenbrock_hess
)

print(result$par)      # Should be near c(1, 1)
print(result$value)   # Should be near 0
```

archopt_advanced_controls

Advanced Control Parameters for archopt

Description

The main [archopt](#) help page documents only the user-facing tolerances and switches (`maxit`, `gtol_abs`, `ftol_abs`, `xtol_abs`, `trace`, `verbose`, `use_qn`). This page documents every other entry that the control list of `archopt()` (and the routed-to `archopt:::archopt_qn()` variant) recognizes, organized by the subsystem each parameter governs.

Cubic regularization

These parameters control the adaptive cubic model $m_k(s) = f_k + g_k^\top s + \frac{1}{2}s^\top H_k s + \frac{\sigma_k}{3}\|s\|^3$ and the sigma adaptation rule (Algorithm 2a/2b of design/pseudocode.qmd).

sigma0 Initial regularization parameter (default 1.0).

sigma_min Floor on sigma_k (default 1e-6). Prevents the cubic term from vanishing entirely on flat regions.

sigma_max Ceiling on sigma_k (default 1e12). Triggers emergency-stop behavior when reached.

eta1 Step-acceptance threshold; steps with $\rho_k \geq \text{eta1}$ are accepted (default 0.1).

eta2 Very-successful threshold; $\rho_k \geq \text{eta2}$ triggers sigma shrinkage (default 0.9).

gamma1 Multiplicative shrink factor on a very-successful step (default 0.5).

gamma2 Multiplicative grow factor on an unsuccessful step (default 2.0).

Trust-region fallback (cubic to TR on flat-ridge detection)

Cubic regularization can stagnate in "flat-ridge" regimes – iterations with the regularization floor pinned, model predictions matching the objective ($\rho \approx 1$), gradient stalling, and a Hessian that is positive-definite but nearly singular. This is outside the local-error-bound condition of Yue, Zhou & So (2018) under which cubic regularization is guaranteed to converge quadratically at degenerate minimizers. arcopt detects the regime and switches once from the cubic subproblem to a trust-region subproblem.

tr_fallback_enabled One-way cubic to TR switch (default TRUE).

tr_fallback_window Sliding-window length for the detector (default 10).

tr_fallback_tol_ridge $\lambda_{\min}(H)$ threshold defining a "near-singular PD" Hessian (default 1e-3).

tr_fallback_rho_tol Tolerance on $|\rho - 1|$ for the "near-perfect model" signal (default 0.1).

tr_fallback_grad_decrease_max Ratio of latest to oldest $\|g\|_{\infty}$ above which the gradient counts as stagnant (default 0.9).

tr_fallback_g_inf_floor Absolute lower bound on $\|g\|_{\infty}$ below which the switch will not fire – keeps the hybrid from triggering at true local minima (default 1e-6).

tr_r0 Initial trust-region radius at the switch (default 1.0).

tr_rmax Maximum trust-region radius (default 100).

tr_eta1 TR step-acceptance threshold (default 0.25).

tr_eta2 TR expansion threshold (default 0.75).

tr_gamma_shrink Radius shrink factor on a poor step (default 0.25).

tr_gamma_grow Radius grow factor on a very-good boundary step (default 2.0).

Quasi-Newton polish (cubic to BFGS line-search)

Once the iterate enters the quadratic attraction basin of a strict local minimum, the cubic regularization penalty has decayed to its floor and contributes negligible damping, but arcopt still evaluates hess() every iteration. For expensive Hessians (analytic AD via Stan, finite differences) this dominates wall-clock time. Polish mode replaces the cubic subproblem with a Wolfe line search along

the BFGS-approximated Newton direction, skipping further `hess()` calls until convergence or until the BFGS approximation drifts.

Off by default in v0.2.0 because the existing manuscript and benchmark problems converge before the five-signal healthy-basin detector accumulates a full window. Enable opt-in for long-running smooth problems with expensive Hessians.

`qn_polish_enabled` Enable the cubic to `qn_polish` bidirectional switch (default FALSE).

`qn_polish_window` Sliding-window length for the healthy-basin detector (default 5).

`qn_polish_rho` Minimum ρ_k required throughout the window (default 0.9).

`qn_polish_lambda_min` Minimum $\lambda_{\min}(H_k)$ required throughout the window (default $1e-3$).

`qn_polish_g_decay` Maximum ratio of consecutive $\|g\|_{\infty}$ values; e.g. 0.5 requires 2x-per-iteration contraction (default 0.5).

`qn_polish_g_inf_floor` Absolute lower bound on $\|g\|_{\infty}$ at window start; prevents firing at convergence (default $1e-8$).

`qn_polish_c1`, `qn_polish_c2` Wolfe line-search constants (defaults $1e-4$ and 0.9).

`qn_polish_alpha_max` Initial step length tried by the line search (default 1.0).

`qn_polish_max_ls_iter` Maximum line-search evaluations per iteration (default 20).

`qn_polish_max_fail` Consecutive line-search failures that trigger a revert to cubic mode (default 3).

`qn_polish_reenter_delay` Cubic iterations required after a revert before `qn_polish` may re-fire (default 5).

`qn_polish_curv_eps` Curvature threshold for skipping BFGS updates to preserve PD (default $1e-10$).

Quasi-Newton variant (`use_qn = TRUE`)

When `control$use_qn = TRUE`, `arcopt()` routes to an internal quasi-Newton variant that approximates H_k via SR1/BFGS updates and does not require a `hess` function. The following parameters apply only when `use_qn = TRUE`.

`qn_method` One of "hybrid" (default), "sr1", "bfgs". "hybrid" uses state-aware routing between SR1-first and BFGS-first orderings based on the current B's eigenstructure and recent ρ values.

`bfgs_tol` Curvature tolerance for the BFGS update (default $1e-10$).

`sr1_skip_tol` SR1 skip-test tolerance (default $1e-8$).

`sr1_restart_threshold` Consecutive SR1 skips before restart (default 5).

`qn_route_demote_rho` ρ below this counts as a "bad" step in the routing FSM (default 0.25).

`qn_route_promote_rho` ρ above this counts as a "good" step (default 0.5).

`qn_route_demote_k` Consecutive bad steps in "pd" routing mode that demote back to "indefinite" (default 2).

`qn_route_promote_k` Consecutive good PD steps in "indefinite" mode that promote to "pd" (default 3).

qn_fd_refresh_k Consecutive bad-rho iterations in "indefinite" mode that trigger an FD-Hessian refresh of B (default 3).

qn_stuck_refresh_k Iterations stuck in "indefinite" mode without promotion that force a refresh (default 100).

use_accel_qn **EXPERIMENTAL**. Enable Nesterov acceleration in the QN path. May improve convergence on strongly convex problems but can hurt nonconvex (default FALSE).

Diagnostics in result\$diagnostics

Mode-dispatch diagnostics are nested under result\$diagnostics so the primary return list stays compact.

solver_mode_final "cubic", "tr", or "qn_polish" – which subproblem solver was active at termination.

ridge_switches Integer count of cubic to TR transitions (0 or 1 in v1; the switch is one-way).

radius_final Final trust-region radius (NA if the solver never switched to TR mode).

qn_polish_switches Integer count of cubic to qn_polish transitions (bidirectional; may be > 1).

qn_polish_reverts Integer count of qn_polish to cubic reversions.

hess_evals_at_polish_switch evaluations\$hess at the first polish switch; compare against final evaluations\$hess to quantify Hessian-evaluation savings.

QN-variant runs add qn_updates, qn_skips, qn_restarts, and qn_fd_refreshes to the same sublist.

See Also

[arcopt](#) for the user-facing entry point.

Index

arcopt, [2](#), [4](#), [7](#)

arcopt_advanced_controls, [4](#), [4](#)