

Package ‘GALAHAD’

May 7, 2026

Type Package

Title Geometry-Adaptive Lyapunov-Assured Hybrid Optimizer with Softplus Reparameterization and Trust-Region Control

Version 2.0.0

Author Richard A. Feiss [aut, cre] (ORCID:
<<https://orcid.org/0009-0008-0409-6042>>)

Maintainer Richard A. Feiss <feiss026@umn.edu>

Description Implements the GALAHAD algorithm (Geometry-Adaptive Lyapunov-Assured Hybrid Optimizer), updated in version 2 to replace the hard-clamp positivity constraint of v1 with a numerically smooth softplus reparameterization, add rho-based trust-region adaptation (actual vs. predicted objective reduction), extend convergence detection to include both absolute and relative function-stall criteria, and enrich the per-iteration history with Armijo backtrack counts and trust-region quality ratios. Parameters constrained to be positive (rates, concentrations, scale parameters) are handled in a transformed z-space via the softplus map so that gradients remain well-defined at the constraint boundary. A two-partition API (positive / euclidean) replaces the three-way T/P/E partition of v1; the legacy form is still accepted for backwards compatibility. Designed for biological modeling problems (germination, dose-response, prion RT-QuIC, survival) where rates, concentrations, and unconstrained coefficients coexist. Developed at the Minnesota Center for Prion Research and Outreach (MNPRO), University of Minnesota. Based on Conn et al. (2000) <[doi:10.1137/1.9780898719857](https://doi.org/10.1137/1.9780898719857)>, Barzilai and Borwein (1988) <[doi:10.1093/imanum/8.1.141](https://doi.org/10.1093/imanum/8.1.141)>, Xu and An (2024) <[doi:10.48550/arXiv.2409.14383](https://doi.org/10.48550/arXiv.2409.14383)>, Polyak (1969) <[doi:10.1016/0041-5553\(69\)90035-4](https://doi.org/10.1016/0041-5553(69)90035-4)>, Nocedal and Wright (2006, ISBN:978-0-387-30303-1), and Dugas et al. (2009) <<https://www.jmlr.org/papers/v10/dugas09a.html>>.

URL <https://github.com/RFeissIV/GALAHAD>

BugReports <https://github.com/RFeissIV/GALAHAD/issues>

Note Development followed an iterative human-machine collaboration where all algorithmic design, statistical methodologies, and biological validation logic were conceptualized, tested, and iteratively refined by Richard A. Feiss through repeated cycles of running experimental data, evaluating analytical outputs, and selecting among candidate algorithms and approaches. AI systems ('Anthropic Claude' and 'OpenAI GPT') served as coding assistants and analytical sounding boards under continuous human direction. The selection of statistical methods, evaluation of biological plausibility, and all final methodology decisions were made by the human author. AI systems did not independently originate algorithms, statistical approaches, or scientific methodologies.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.2.0)

Imports stats, utils

Suggests testthat (>= 3.0.0)

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

BuildResaveData true

Repository CRAN

Date/Publication 2026-03-08 06:11:20 UTC

Contents

GALAHAD	2
galahad_numgrad	6
galahad_parts	7

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

GALAHAD	<i>GALAHAD: Geometry-Adaptive Optimizer with Softplus Reparameterization</i>
---------	--

Description

Minimises a smooth objective $V(\theta)$ over a mixed-geometry parameter space. Parameters declared as **positive** are handled via a softplus reparameterization so that positivity is guaranteed analytically and gradients remain smooth at the constraint boundary. Parameters declared as **euclidean** are unconstrained.

The algorithm combines:

- Diagonal Hessian preconditioning via a secant-based exponential moving average.
- Adaptive step-size selection: Polyak (when V_star is supplied) -> Barzilai-Borwein -> constant default.
- Armijo backtracking line-search with a configurable monotone or rho-accept mode.
- Trust-region projection with rho-based radius adaptation (actual vs. predicted reduction).
- Dual convergence criterion: gradient + step tolerance (primary) and absolute or relative function-stall over five iterations (secondary).

Usage

```
GALAHAD(V, gradV, theta0, parts, control = list(), callback = NULL)
```

Arguments

<code>V</code>	Objective function: <code>function(theta) -> scalar</code> . Must return a finite numeric scalar for any <code>theta</code> encountered during optimization.
<code>gradV</code>	Gradient function: <code>function(theta) -> numeric vector of length p</code> . Must return finite values. A finite-difference approximation (e.g., <code>galahad_numgrad</code>) may be substituted when an analytical gradient is unavailable.
<code>theta0</code>	Initial parameter vector, numeric of length p . Elements indexed by <code>parts\$positive</code> must be strictly positive.
<code>parts</code>	Named list specifying the parameter partition. See section <i>Parameter partitions</i> above.
<code>control</code>	Optional named list of control parameters. See section <i>Control parameters</i> above.
<code>callback</code>	Optional function called at the end of each iteration. Receives a list with elements <code>iter</code> (integer), <code>theta</code> (numeric vector), <code>value</code> (scalar), and <code>grad_norm</code> (scalar). Errors inside the callback are silently suppressed.

Value

A named list with the following elements:

<code>theta</code>	Numeric vector; final parameter estimate in the original θ space (<code>theta[positive] > 0</code> is guaranteed).
<code>value</code>	Scalar; $V(\theta)$ at the final iterate.
<code>grad_inf</code>	Sup-norm of the gradient at the final iterate.
<code>converged</code>	Logical; TRUE if any convergence criterion was satisfied before <code>max_iter</code> .
<code>status</code>	Character; "converged" or "max_iter".
<code>reason</code>	Character; one of "GRAD_TOL", "FUNC_STALL_ABS", "FUNC_STALL_REL", "MAX_ITER".
<code>iterations</code>	Integer; number of iterations performed.
<code>history</code>	A data.frame with one row per iteration and columns <code>iter</code> , <code>f</code> , <code>g_inf</code> , <code>step_norm</code> , <code>df</code> (change in V), <code>eta</code> , <code>method</code> (step-size method: "POLYAK", "BB", or "DEFAULT"), <code>armijo_iters</code> , <code>pred_red</code> (predicted reduction), and <code>rho</code> (actual / predicted reduction).
<code>diagnostics</code>	List with <code>mean_L_hat</code> (mean diagonal Hessian estimate at termination), <code>final_delta</code> (final trust radius), <code>enforce_monotone</code> , <code>use_rho_accept</code> , and <code>parameterization</code> ("softplus").
<code>certificate</code>	List with <code>converged</code> and <code>reason</code> ; a compact convergence certificate.

Parameter partitions

The parts argument partitions the indices 1:p (where $p = \text{length}(\text{theta0})$) into two groups.

Preferred form:

positive Integer indices of parameters constrained to $(0, \infty)$. Typical examples: rate constants, concentrations, standard deviations, Hill coefficients. Internally reparameterized via $\theta_i = \text{softplus}(z_i)$.

euclidean Integer indices of unconstrained parameters. Typical examples: location parameters, regression coefficients, log-EC50.

Legacy form (v1 compatibility): `list(T = ..., P = ..., E = ...)` is also accepted. Indices in T and P are mapped to **positive**; indices in E are mapped to **euclidean**.

All indices must appear exactly once and cover 1:p.

Control parameters

Pass as named elements of the control list. Any omitted element takes the default shown below.

max_iter Maximum iterations (default: 2000).

tol_g Convergence: sup-norm of gradient $\leq \text{tol}_g$ (default: 1e-6).

tol_x Convergence: step norm $\leq \text{tol}_x$ (default: 1e-9).

tol_f Stall: absolute change in V over last five iterations $\leq \text{tol}_f$ (default: 1e-12).

tol_f_rel Stall: relative change in V over last five iterations $\leq \text{tol}_f_rel$ (default: 1e-9).

delta Initial trust-region radius (default: 1.0).

delta_min Minimum trust-region radius (default: 1e-6).

delta_max Maximum trust-region radius (default: 1e3).

eta0 Initial / fallback step size (default: 1.0).

eta_min Minimum step size (default: 1e-8).

eta_max Maximum step size (default: 10.0).

c1 Armijo sufficient-decrease constant (default: 1e-4).

armijo_max Maximum Armijo backtrack steps (default: 20).

L_est Initial diagonal Hessian estimate (default: 1.0).

l_min, l_max Bounds for Hessian diagonal entries (defaults: 1e-8, 1e8).

V_star Known or estimated minimum value of V , used to compute Polyak step sizes. NULL disables Polyak steps (default: NULL).

lambda L2 regularization coefficient; adds $\lambda \|\theta\|^2$ to the objective (default: 0).

enforce_monotone Logical; if TRUE (default), the Armijo condition must be satisfied. Set to FALSE together with `use_rho_accept = TRUE` to use rho-accept mode instead.

use_rho_accept Logical; if TRUE, accept a step when $\rho \geq \text{rho_accept}$ rather than requiring Armijo decrease (default: FALSE).

rho_accept Minimum acceptable rho for step acceptance in rho-accept mode (default: 0.1).

rho_shrink Trust-radius shrinks when $\rho < \text{rho_shrink}$ (default: 0.25).

rho_grow Trust-radius grows when $\rho >$ rho_grow and step fills the region (default: 0.75).
 shrink_factor Multiplicative shrink factor for trust radius (default: 0.5).
 grow_factor Multiplicative growth factor for trust radius (default: 1.5).
 softplus_cap Upper threshold in the softplus overflow guard (default: 700). Rarely needs changing.

References

- Barzilai, J., & Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1), 141–148. doi:10.1093/imanum/8.1.141
- Conn, A. R., Gould, N. I. M., & Toint, P. L. (2000). *Trust-Region Methods*. SIAM. doi:10.1137/1.9780898719857
- Dugas, C., Bengio, Y., Belisle, F., Nadeau, C., & Garcia, R. (2009). Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research*, 10(42), 1239–1262. <https://www.jmlr.org/papers/v10/dugas09a.html>
- Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization* (2nd ed.). Springer. ISBN 978-0-387-30303-1.
- Polyak, B. T. (1969). The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, 9(4), 94–112. doi:10.1016/00415553(69)900354
- Xu, X., & An, C. (2024). A trust region method with regularized Barzilai-Borwein step-size for large-scale unconstrained optimization. *arXiv preprint*. doi:10.48550/arXiv.2409.14383

See Also

[galahad_numgrad](#) for finite-difference gradient approximation; [galahad_parts](#) for a helper that builds the parts list.

Examples

```
## -- Example 1: purely positive parameters (exponential decay) -----
set.seed(1)
t <- seq(0, 5, by = 0.5)
y <- 3 * exp(-0.8 * t) + rnorm(length(t), sd = 0.05)

obj <- function(theta) sum((y - theta[1] * exp(-theta[2] * t))^2)
grd <- function(theta) {
  r <- y - theta[1] * exp(-theta[2] * t)
  dA <- -2 * sum(r * exp(-theta[2] * t))
  dk <- -2 * sum(r * (-t) * theta[1] * exp(-theta[2] * t))
  c(dA, dk)
}

fit <- GALAHAD(V      = obj,
              gradV   = grd,
              theta0  = c(A = 2, k = 0.5),
              parts   = list(positive = c(1L, 2L),
                             euclidean = integer(0)))
fit$theta          # close to c(3, 0.8)
```

```

fit$converged
fit$reason

## -- Example 2: mixed geometry (4PL dose-response) -----

set.seed(42)
dose <- c(0.01, 0.1, 1, 5, 10, 50, 100)
y4pl <- 10 + (90 - 10) / (1 + exp(-1.5 * (log(dose) - log(5)))) +
  rnorm(length(dose), sd = 2)

## theta = c(bottom, hill, logEC50, top)
## hill > 0 is positive; bottom, logEC50, top are unconstrained
obj4 <- function(th) {
  pred <- th[1] + (th[4] - th[1]) / (1 + exp(-th[2] * (log(dose) - th[3])))
  sum((y4pl - pred)^2)
}
grd4 <- function(th) galahad_numgrad(obj4, th)

fit4 <- GALAHAD(V      = obj4,
               gradV   = grd4,
               theta0  = c(bottom = 15, hill = 1, logEC50 = log(5), top = 80),
               parts   = list(positive = 2L,          # hill must be > 0
                              euclidean = c(1L, 3L, 4L)))

fit4$theta
fit4$converged

## -- Example 3: legacy v1 partition syntax (backwards compatible) -----

set.seed(7)
p <- 10
theta_true <- abs(rnorm(p)) + 0.5
V_q <- function(th) sum((th - theta_true)^2)
gV_q <- function(th) 2 * (th - theta_true)

## Old-style T / P / E partition -- still accepted in v2
fit_legacy <- GALAHAD(V      = V_q,
                    gradV   = gV_q,
                    theta0  = rep(1, p),
                    parts   = list(T = 1:3, P = 4:7, E = 8:10))

fit_legacy$converged

```

Description

Computes a central-difference numerical gradient of f at θ . Useful when an analytical gradient is not available.

Usage

```
galahad_numgrad(f, theta, eps = 1e-07)
```

Arguments

f	Scalar-valued function of a numeric vector.
theta	Numeric vector at which to evaluate the gradient.
eps	Step size scaling factor (default 1e-7). The actual step for parameter i is $\text{eps} * (\text{theta}[i] + 1)$.

Value

Numeric gradient vector of the same length as theta.

Examples

```
f <- function(th) sum((th - c(2, 3))^2)
galahad_numgrad(f, c(0, 0)) # should be close to c(-4, -6)
```

galahad_parts	<i>Build a GALAHAD parts list</i>
---------------	-----------------------------------

Description

Convenience constructor that validates and assembles the parts argument for [GALAHAD](#).

Usage

```
galahad_parts(positive = integer(0), euclidean = integer(0), p = NULL)
```

Arguments

positive	Integer vector of indices (1-based) for parameters that must be positive. May be <code>integer(0)</code> if there are no positive constraints.
euclidean	Integer vector of indices for unconstrained parameters. May be <code>integer(0)</code> .
p	Total number of parameters. If supplied, indices are validated against 1:p.

Value

A named list `list(positive = ..., euclidean = ...)`.

Examples

```
galahad_parts(positive = c(1L, 2L), euclidean = 3L, p = 3)
```

Index

GALAHAD, [2](#), [7](#)

galahad_numgrad, [3](#), [5](#), [6](#)

galahad_parts, [5](#), [7](#)