Package ‘taoR’

February 1, 2016

Type Package

Title TAO Bindings for R

Version 0.1

Date 2015-12-19

Author Jan Tilly and Nick Janetos

Maintainer Jan Tilly <jantilly@gmail.com>

URL https://github.com/jtilly/taoR/

Description Provides Toolkit for Advanced Optimization (TAO) bindings for R using Rcpp. This package requires a working installation of the Portable, Extensible Toolkit for Scientific Computation (PETSc).

License GPL-2

LazyData TRUE

Imports Rcpp

LinkingTo Rcpp

Suggests testthat

SystemRequirements Portable, Extensible Toolkit for Scientific Computation (PETSc) libraries. This package attempts to install PETSc during the build step if not already available on the system.

OS_type unix

RoxygenNote 5.0.1

R topics documented:

tao *R bindings for the TAO optimization library.*

Description

Various optimization routines from the TAO optimization library. See the TAO documentation for a complete listing.

Usage

```
tao(par, fn, gr = NULL, hs = NULL, method = c("lmvm", "nls", "ntr", "ntl",
  "cg", "tron", "blmvm", "gpcg", "nm", "pounders"), control = list(),
  n = NULL, lb = NULL, ub = NULL)
```

Arguments

`par` Initial values for the parameters to be optimized over.

`fn` A function to be minimized (or maximized), with first argument the vector of parameters over which minimization is to take place. It should return a scalar result.

`gr` A function to return the gradient, if using a gradient-based optimization method.

`hs` A function to return the hessian, if using an algorithm which uses the hessian.

`method` The method to be used. See 'Details'.

`control` A list of control parameters. See 'Details'.

`n` The number of elements of objfun (optional).

`lb` A vector with lower variable bounds (optional)

`ub` A vector with upper variable bounds (optional)

Value

A list with final parameter values, the objective function, and information on why the optimizer stopped.

Examples

```
# Gradient-free method
objfun = function(x) c((x[1] - 3), (x[2] + 1))
ret = tao(c(1, 2),
              objfun,
              method = "pounders",
              control = list(tao_pounders_delta=0.1))
ret$x

# Gradient-based method: Limited memory variable metric method
objfun = function(x) (x[1] - 3)^2 + (x[2] + 1)^2
grafun = function(x) c(2*(x[1] - 3), 2*(x[2] + 1))
```

```
ret = tao(c(1, 2),
                objfun,
                gr = grafun,
                method = "lmvm")
ret$x

# Gradient-based method: Limited memory variable metric method with bounds
objfun = function(x) (x[1] - 3)^2 + (x[2] + 1)^2
grafun = function(x) c(2*(x[1] - 3), 2*(x[2] + 1))
inequal = function(x) c(x[1] - 2, x[2] - 2)

ret = tao(c(1, 2),
                objfun,
                gr = grafun,
                method = "blmvm")
ret$x

# Hessian (Newton Trust Region)
objfun = function(x) (x[1] - 3)^2 + (x[2] + 1)^2
grafun = function(x) c(2*(x[1] - 3), 2*(x[2] + 1))
hesfun = function(x) matrix(c(2, 0, 0, 2), nrow = 2, ncol = 2)

ret = tao(c(1, 2),
                objfun,
                gr = grafun,
                hs = hesfun,
                method = "ntr")
ret$x
```

tao_cpp *Use TAO to minimize an objective function*

Description

tao_cpp is an internal function of this package. It is recommended that users call tao instead, which has a more convenient syntax and performs thorough input checking.

Usage

```
tao_cpp(functions, start_values, method, options, n, lower_bounds, upper_bounds)
```

Arguments

functions	is a list of Rcpp functions. The first is always the objective function. The second and third are optionally the Jacobian and the Hessian functions.
start_values	is a vector containing the starting values of the parameters.
method	is a string that determines the type of optimizer to be used.
options	is a list containing option values for the optimizer

`n` is the number of elements in the objective function.

`lower_bounds` is a vector with lower bounds

`upper_bounds` is a vector with upper bounds

Value

a list with the objective function and the final parameter values

Examples

```
# use pounders
objfun = function(x) c((x[1] - 3), (x[2] + 1))
ret = tao_cpp(functions = list(objfun = objfun),
              start_values = c(1, 2),
              method = "pounders",
              options = list(),
              n = 2,
              lower_bounds = c(-2, -2),
              upper_bounds = c(5, 5))
ret$x

# use Nelder-Mead
objfun = function(x) sum(c((x[1] - 3)^2, (x[2] + 1))^2)
ret = tao_cpp(functions = list(objfun = objfun),
                  start_values = c(1, 2),
                  method = "nm",
                  options = list(),
                  n = 1,
                  lower_bounds = c(-2, -2),
                  upper_bounds = c(5, 5))
ret$x
```

tao_finalize *Finalize TAO*

Description

This function is called automatically when the package is unloaded.

Usage

```
tao_finalize()
```

`tao_init` *Initialize TAO*

Description

This function is called automatically when the package is loaded.

Usage

```
tao_init()
```

Index