Package ‘neldermead’

July 2, 2014

Type Package

Title R port of the Scilab neldermead module

Version 1.0-9

Date 2014-03-02

Author Sebastien Bihorel, Michael Baudin (author of the original module)

Maintainer Sebastien Bihorel <sb.pmlab@gmail.com>

Depends optimbase (>= 1.0-9), optimsimplex (>= 1.0-5), methods

Description Provides several direct search optimization algorithms based on the simplex method. The provided algorithms are direct search algorithms, i.e. algorithms which do not use the derivative of the cost function. They are based on the update of a simplex. The following algorithms are available: the fixed shape simplex method of Spendley, Hext and Himsworth (unconstrained optimization with a fixed shape simplex), the variable shape simplex method of Nelder and Mead (unconstrained optimization with a variable shape simplex made), and Box's complex method (constrained optimization with a variable shape simplex).

License CeCILL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-03 00:10:02

R topics documented:

  neldermead-package ........................................ 2
costf.transpose ........................................... 4
fmin.gridsearch ............................................. 5
fminbnd .................................................... 6
Description

The goal of this package is to provide a Nelder-Mead direct search optimization method. That Nelder-Mead algorithm may be used in the following optimization context:

- there is no need to provide the derivatives of the objective function,
- the number of parameters is small (up to 10-20),
- there are bounds and/or non linear constraints.

Design

This package provides the following components:

- **neldermead** provides various Nelder-Mead variants and manages for Nelder-Mead specific settings, such as the method to compute the initial simplex, the specific termination criteria,
- **fminsearch** provides a simplified Nelder-Mead algorithm. Specific termination criteria, initial simplex and auxiliary settings are automatically configured.
- **fminbnd** provides a simplified Box algorithm, ie the equivalent of **fminsearch** for unconstrained search.
- **optimset**, **optimget** provide commands to emulate their Scilab counterparts.
- **optimplotfunccount**, **optimplotx** and **optimplotfval** provide plotting features for the **fminsearch** function (Not implemented yet).
- **nmplot** provides a high-level component which provides directly output pictures for Nelder-Mead algorithm. (Not implemented yet).
The current component is based on the following packages

- **optimbase**: provides an abstract class for a general optimization component, including the number of variables, the minimum and maximum bounds, the number of non linear inequality constraints, the loggin system, various termination criteria, the cost function, etc...

- **optimsimplex**: provides a class to manage a simplex made of an arbitrary number of vertices, including the computation of a simplex by various methods (axes, regular, Pfeffer’s, randomized bounds), the computation of the size by various methods (diameter, sigma+, sigma-, etc...).

**Features**

The following is a list of features the Nelder-Mead prototype algorithm currently provides:

- Provides 3 algorithms, including
  - the fixed shape algorithm of Spendley et al.,
  - the variable shape algorithm of Nelder and Mead,
  - Box’s ‘complex’ algorithm managing bounds and nonlinear inequality constraints based on arbitrary number of vertices in the simplex.

- Manage various simplex initializations:
  - initial simplex given by user,
  - initial simplex computed with a length and along the coordinate axes,
  - initial regular simplex computed with formula of Spendley et al.,
  - initial simplex computed by a small perturbation around the initial guess point.

- Manage cost function:
  - optional additional argument,
  - direct communication of the task to perform: cost function or inequality constraints.

- Manage various termination criteria, including maximum number of iterations, tolerance on function value (relative or absolute):
  - tolerance on x (relative or absolute),
  - tolerance on standard deviation of function value (original termination criteria in Box 1965),
  - maximum number of evaluations of cost function,
  - absolute or relative simplex size.

- Manage the history of the convergence, including:
  - history of function values,
  - history of optimum point,
  - history of simplices,
  - history of termination criteria.

- Provide a plot command which allows to graphically see the history of the simplices toward the optimum (Not yet implemented).

- Provide query features for the status of the optimization process: number of iterations, number of function evaluations, status of execution, function value at initial point, function value at optimal point, etc...

- Kelley restart based on simplex gradient.

- O’Neill restart based on factorial search around optimum.
Details

Package: neldermead
Type: Package
Version: 1.0-9
Date: 2014-03-02
License: CeCILL-2
LazyLoad: yes

See vignette('neldermead', package='neldermead') for more information.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

References

'Detection and Remediation of Stagnation in the Nelder–Mead Algorithm Using a Sufficient Decrease Condition', Kelley C. T., SIAM J. on Optimization, 1999

See Also

optimbase optimsimplex

costf.transposex    Cost Function Call

Description

Call the cost function after transposition of the value of the point estimate x, so that the input row vector, given by optimsimplex, is transposed into a column vector as required by the cost function.
**fmin.gridsearch**

**Usage**

```r
costf.transpose(x = NULL, this = NULL)
```

**Arguments**

- `x` The point estimate provide as a row matrix.
- `this` A neldermead object.

**Value**

Return the value of the cost function (called by `neldermead.costf`).

**Author(s)**

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

**See Also**

- `neldermead.costf`

---

**fmin.gridsearch**  
*Grid evaluation of an unconstrained cost function*

**Description**

Evaluate an unconstrained cost function on a grid of points around a given initial point estimate.

**Usage**

```r
fmin.gridsearch(fun = NULL, x0 = NULL, xmin = NULL, 
                 xmax = NULL, npts = 3, alpha = 10)
```

**Arguments**

- `fun` An unconstrained cost function returning a numeric scalar, similar to those used in the `fminsearch` function.
- `x0` The initial point estimate, provided as a numeric vector.
- `xmin` Optional: a vector of lower bounds.
- `xmax` Optional: a vector of upper bounds.
- `npts` An integer scalar greater than 2, indicating the number of evaluation points will be used on each dimension to build the search grid.
- `alpha` A vector of numbers greater than 1, which give the factor(s) used to calculate the evaluation range of each dimension of the search grid (see Details). If `alpha` length is lower than that of `x0`, elements of `alpha` are recycled. If its length is higher than that of `x0`, `alpha` is truncated.
Details

fmin.gridsearch evaluates the cost function at each point of a grid of \( npts \times \text{length}(x0) \) points. If lower (xmin) and upper (xmax) bounds are provided, the range of evaluation points is limited by those bounds and alpha is not used. Otherwise, the range of evaluation points is defined as \([x0/alpha, x0*alpha]\).

The actual evaluation of the cost function is delegated to optimbase.gridsearch.

Value

Return a data.frame with the coordinates of the evaluation point, the value of the cost function and its feasibility. Because the cost function is unconstrained, it is always feasible. The data.frame is ordered by feasibility and increasing value of the cost function.

Author(s)

Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

fminsearch, optimbase.gridsearch

Description

EXPERIMENTAL.

This function searches for the constrained minimum of a given cost function. The provided algorithm is a direct search algorithm, i.e. an algorithm which does not use the derivative of the cost function. It is based on the update of a simplex, which is a set of \( k=n+1 \) vertices, where each vertex is associated with one point, which coordinates are constrained within user-defined boundaries, and with one function value. This algorithm corresponds to a version of the Box algorithm, based on bounds and no non-linear constraints. This function is based on a specialized use of the more general neldermead function bundle. Users who want to have a more flexible solution based on direct search algorithms should consider using the neldermead functions instead of the fminbnd function.

Usage

fminbnd(fun=NULL, x0=NULL, xmin=NULL, xmax=NULL, options=NULL, verbose=FALSE)
Arguments

fun A cost function return a numeric scalar.

x0 A numerical vector of initial guesses (length n).

xmin A numerical vector of lower bounds for x0 (length n).

xmax A numerical vector of upper bounds for x0 (length n).

options A list of optimization options, which drives the behaviour of fminbnd. These options must be set with the optimset function (see ?optimset) which returns a list with the following elements:

MaxIter The maximum number of iterations. The default is 200 * n.

MaxFunEvals The maximum number of evaluations of the cost function. The default is 200 * n.

BoxTolFun The absolute tolerance on function value. The default value is 1.e-4.

TolFun The absolute tolerance on function value. The default value is 1.e-4.

TolX The absolute tolerance on simplex size. The default value is 1.e-4.

Display The verbose level.

OutputFcn The output function, or a list of output functions called at the end of each iteration. The default value is NULL.

PlotFcns The plot function, or a list of plotput functions called at the end of each iteration. The default value is empty.

verbose The verbose option, controlling the amount of messages.

Details

Termination criteria

In this section, we describe the termination criteria used by fminbnd. The criteria is based on the following variables:

boxkount the current number of time the tolerance on the cost function was met, and

shiftfv the absolute value of the difference of function value between the highest and lowest vertices.

If both shiftfv < options$TolFun and boxkount < options$nbMatch conditions are true, then the iterations stop.

The initial simplex

The fminbnd algorithm uses a special initial simplex, which is an heuristic depending on the initial guess. The strategy chosen by fminbnd corresponds to the content of simplex@method element of the neldermead object (set to ’randbounds’). It is applied using the content of the bounds@min and bounds@max elements to generate a simplex with random vertices within the boundaries defined by the user (ie, xmin, and xmax). This method is an heuristic which is presented in ’A New Method of Constrained Optimization and a Comparison With Other Methods’ by M.J. Box. See in the help of optimsimplex for more details.

The number of iterations

In this section, we present the default values for the number of iterations in fminbnd.
The `options` input argument is an optional list which can contain the `MaxIter` field, which stores the maximum number of iterations. The default value is 200n, where n is the number of variables. The factor 200 has not been chosen by chance, but is the result of experiments performed against quadratic functions with increasing space dimension. This result is presented in ‘Effect of dimensionality on the Nelder-Mead simplex method’ by Lixing Han and Michael Neumann. This paper is based on Lixing Han’s PhD, ‘Algorithms in Unconstrained Optimization’. The study is based on numerical experiments with a quadratic function where the number of terms depends on the dimension of the space (i.e. the number of variables). Their study showed that the number of iterations required to reach the tolerance criteria is roughly 100n. Most iterations are based on inside contractions. Since each step of the Nelder-Mead algorithm only require one or two function evaluations, the number of required function evaluations in this experiment is also roughly 100n.

**Output and plot functions**

The `optimset` function can be used to configure one or more output and plot functions. The output or plot function is expected to have the following definition:

```r
myfun <- function(x, optimValues, state)
```

The input arguments `x`, `optimValues` and `state` are described in detail in the `optimset` help page. The `optimValues$procedure` field represents the type of step performed at the current iteration and can be equal to one of the following strings:

- " (the empty string),
- 'initial simplex',
- 'reflect (Box)'.

**Value**

Return a object of class `neldermead`. Use the `neldermead.get` to extract the following element from the returned object:

- **xopt** The vector of n numeric values, minimizing the cost function.
- **fopt** The minimum value of the cost function.
- **exitflag** The flag associated with exist status of the algorithm. The following values are available:
  - -1 The maximum number of iterations has been reached.
  - 0 The maximum number of function evaluations has been reached.
  - 1 The tolerance on the simplex size and function value delta has been reached. This signifies that the algorithm has converged, probably to a solution of the problem.

- **output** A list which stores detailed information about the exit of the algorithm. This list contains the following fields:
  - **algorithm** A string containing the definition of the algorithm used, i.e. ‘Nelder-Mead simplex direct search’.
  - **funcCount** The number of function evaluations.
  - **iterations** The number of iterations.
  - **message** A string containing a termination message.

**Author(s)**

Author: Sebastien Bihorel (<sb.pmlab@gmail.com>)
Examples

In the following example, we use the fminbnd function to compute the minimum of a quadratic function. We first define the function ‘quad’, and then use the fminbnd function to search the minimum, starting with the initial guess (1.2, 1.9) and bounds of (1, 1) and (2, 2). In this particular case, 11 iterations are performed with 10 function evaluations.

```r
quad <- function(x){
  y <- x[1]^2 + x[2]^2
}
sol <- fminbnd(quad,c(1.2,1.9),c(1,1),c(2,2))
summary(sol)
```

Description

This function calls the cost function and makes it match neldermead requirements. It is used in the fminbnd function as the function element of the neldermead object (see ?neldermead and ?neldermead.set).

Usage

```r
fminbnd.function(x = NULL, index = NULL, fmsfundata = NULL)
```

Arguments

- `x`: A single column vector of parameter estimates.
- `index`: An integer variable set to 2, indicating that only the cost function is to be computed by the algorithm.
- `fmsfundata`: An object of class 'optimbase.functionargs' and with (at least) a `fun` element, which contains the user-defined cost function.

Value

Returns a list with the following elements:

- `f`: The value of the cost function at the current point estimate.
- `index`: The same index variable.
- `this`: A list with a single element `costargument` which contains `fmsfundata`. 

See Also

optimset
fminbnd.outputfun

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

fminbnd, neldermead, neldermead.set,

fminbnd.outputfun fminbnd Output Function Call

Description

This function calls the output function and make it match neldermead requirements. It is used in the fminbnd function as the outputcommand element of the neldermead object (see ?neldermead and ?neldermead.set).

Usage

fminbnd.outputfun(state = NULL, data = NULL, fmsdata = NULL)

Arguments

state The current state of the algorithm either 'init', 'iter' or 'done'.
data The data at the current state. This is an object of class 'neldermead.data', i.e. a list with the following elements:
x The current parameter estimates.
fval The current value of the cost function.
simplex The current simplex object.
itration The number of iterations performed.
fucccount The number of function evaluations.
step The type of step in the previous iteration.
fmsdata This is an object of class 'optimbase.functionargs' which contains specific data of the fminbnd algorithm:
  Display what to display
  OutputFen the array of output functions
  PlotFcns the array of plot functions

Value

This function does not return any data, but execute the output function(s).
Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

fminbnd, neldermead, neldermead.set.

Description

This function searches for the unconstrained minimum of a given cost function. The provided algorithm is a direct search algorithm, i.e. an algorithm which does not use the derivative of the cost function. It is based on the update of a simplex, which is a set of k>=n+1 vertices, where each vertex is associated with one point and one function value. This algorithm is the Nelder-Mead algorithm. This function is based on a specialized use of the more general neldermead function bundle. Users who want to have a more flexible solution based on direct search algorithms should consider using the neldermead functions instead of the fminsearch function.

Usage

fminsearch(fun = NULL, x0 = NULL, options = NULL, verbose=FALSE)

Arguments

fun A cost function return a numeric scalar.
x0 A numerical vector of initial guesses (length n).
options A list of optimization options, which drives the behaviour of fminsearch. These options must be set with the optimset function (see ?optimset) which returns a list with the following elements:

MaxIter The maximum number of iterations. The default is 200 * n.
MaxFunEvals The maximum number of evaluations of the cost function. The default is 200 * n.
TolFun The absolute tolerance on function value. The default value is 1.e-4.
TolX The absolute tolerance on simplex size. The default value is 1.e-4.
Display The verbose level.
OutputFcn The output function, or a list of output functions called at the end of each iteration. The default value is NULL.
PlotFcns The plot function, or a list of plotput functions called at the end of each iteration. The default value is empty.

verbose The verbose option, controlling the amount of messages.
Details

Termination criteria

In this section, we describe the termination criteria used by fminsearch. The criteria is based on the following variables:

- ssize the current simplex size,
- shiftfv the absolute value of the difference of function value between the highest and lowest vertices.

If both ssize < options$TolX and shiftfv < options$TolFun conditions are true, then the iterations stop. The size of the simplex is computed using the 'sigmaplus' method of the optimsimplex package. The 'sigmaplus' size is the maximum length of the vector from each vertex to the first vertex. It requires one loop over the vertices of the simplex.

The initial simplex

The fminsearch algorithm uses a special initial simplex, which is an heuristic depending on the initial guess. The strategy chosen by fminsearch corresponds to the content of simplex$method element of the neldermead object (set to 'pfeffer'). It is applied using the content of the simplex$deltausual (0.05) and simplex$deltazero (0.0075) elements. Pfeffer's method is an heuristic which is presented in 'Global Optimization Of Lennard-Jones Atomic Clusters' by Ellen Fan. It is due to L. Pfeffer at Stanford. See in the help of optimsimplex for more details.

The number of iterations

In this section, we present the default values for the number of iterations in fminsearch.

The options input argument is an optional list which can contain the MaxIter field, which stores the maximum number of iterations. The default value is 200n, where n is the number of variables. The factor 200 has not been chosen by chance, but is the result of experiments performed against quadratic functions with increasing space dimension. This result is presented in 'Effect of dimensionality on the Nelder-mead simplex method' by Lixing Han and Michael Neumann. This paper is based on Lixing Han’s PhD, 'Algorithms in Unconstrained Optimization'. The study is based on numerical experiments with a quadratic function where the number of terms depends on the dimension of the space (i.e. the number of variables). Their study showed that the number of iterations required to reach the tolerance criteria is roughly 100n. Most iterations are based on inside contractions. Since each step of the Nelder-Mead algorithm only require one or two function evaluations, the number of required function evaluations in this experiment is also roughly 100n.

Output and plot functions

The optimset function can be used to configure one or more output and plot functions. The output or plot function is expected to have the following definition:

myfun <- function(x, optimValues, state)

The input arguments x, optimValues and state are described in detail in the optimset help page. The optimValues$procedure field represents the type of step performed at the current iteration and can be equal to one of the following strings:

- "" (the empty string),
- 'initial simplex',
- 'expand',
- 'reflect',
- 'contract'.
Value

Return a object of class neldermead. Use the neldermead.get to extract the following element from the returned object:

\textbf{xopt} The vector of n numeric values, minimizing the cost function.

\textbf{fopt} The minimum value of the cost function.

\textbf{exitflag} The flag associated with exist status of the algorithm. The following values are available:

\begin{itemize}
  \item {-1} The maximum number of iterations has been reached.
  \item {0} The maximum number of function evaluations has been reached.
  \item {1} The tolerance on the simplex size and function value delta has been reached. This signifies that the algorithm has converged, probably to a solution of the problem.
\end{itemize}

\textbf{output} A list which stores detailed information about the exit of the algorithm. This list contains the following fields:

\begin{itemize}
  \item \textbf{algorithm} A string containing the definition of the algorithm used, i.e. ‘Nelder-Mead simplex direct search’.
  \item \textbf{funcCount} The number of function evaluations.
  \item \textbf{iterations} The number of iterations.
  \item \textbf{message} A string containing a termination message.
\end{itemize}

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)

Author of R adaptation: Sebastien Bihorel (sb.pmlab@gmail.com)

References


See Also

optimset neldermead

Examples

In the following example, we use the fminsearch function to compute the minimum
of the Rosenbrock function. We first define the function 'banana', and then use
the fminsearch function to search the minimum, starting with the initial guess
(-1.2, 1.0). In this particular case, 85 iterations are performed with 159
function evaluations.

```r
banana <- function(x)
  y <- 100*(x[2]-x[1]^2)^2 + (1-x[1])^2
sol <- fminsearch(banana, c(-1.2,1))
sol
```

In the following example, we configure the absolute tolerance on the size of
the simplex to a larger value, so that the algorithm performs less iterations.
Since the default value of 'TolX' for the fminsearch function is 1.e-4, we
decide to use 1.e-2. The optimset function is used to create an optimization
option list and the field 'TolX' is set to 1.e-2. The options list is then
passed to the fminsearch function as the third input argument. In this
particular case, the number of iterations is 70 with 130 function evaluations.

```r
opt <- optimset(tolx=1.e-2)
sol <- fminsearch(banana, c(-1.2,1), opt)
sol
```

In the following example, we want to produce intermediate outputs of the
algorithm. We define the outfun function, which takes the current point x as
input argument. The function plots the current point into the current graphic
window with the plot function. We use the 'OutputFcn' feature of the optimset
function and set it to the output function. Then the option list is passed
to the fminsearch function. At each iteration, the output function is called
back, which creates and update a plot. While this example creates a 2D plot,
the user may customized the output function so that it writes a message in
the console, write some data into a data file, etc... The user can distinguish
between the output function (associated with the 'OutputFcn' option) and the
plot function (associated with the 'PlotFcns' option). See the optimset for
more details on this feature.

```r
outfun <- function(x, optimValues, state){
  plot(x[1],x[2],xlim=c(-1.5,1.5),ylim=c(-1.5,1.5))
  par(new=TRUE)
}
opt <- optimset(OutputFcn=outfun)
sol <- fminsearch(banana, c(-1.2,1), opt)
sol
```

The 'Display' option allows to get some input about the intermediate steps of
the algorithm as well as to be warned in case of a convergence problem.
In the following example, we present what happens in case of a convergence
fminsearch.function

#problem. We set the number of iterations to 10, instead of the default 400
#iterations. We know that 85 iterations are required to reach the convergence
#criteria. Therefore, the convergence criteria is not met and the maximum number
#of iterations is reached.

```
opt <- optimset(MaxIter=10)
sol <- fminsearch(banana, c(-1.2,1), opt)
```

#Since the default value of the 'Display' option is 'notify', a message is
#generated, which warns the user about a possible convergence problem. The
#previous script produces the following output.
# Exiting: Maximum number of iterations has been exceeded
#  - increase MaxIter option.
#  Current function value: 4.1355598

#In the following example, we present how to display intermediate steps used by
#the algorithm. We simply set the 'Display' option to the 'iter' value. This
#option allows to see the number of function evaluations, the minimum function
#value and which type of simplex step is used for the iteration.
```
opt <- optimset(Display='iter')
sol <- fminsearch(banana, c(-1.2,1), opt)
sol
```

---

fminsearch.function  

fminsearch Cost Function Call

Description

This function calls the cost function and makes it match neldermead requirements. It is used in the fminsearch function as the function element of the neldermead object (see ?neldermead and ?neldermead.set).

Usage

```
fminsearch.function(x = NULL, index = NULL, fmsfundata = NULL)
```

Arguments

- **x**
  - A single column vector of parameter estimates.
- **index**
  - An integer variable set to 2, indicating that only the cost function is to be computed by the algorithm.
- **fmsfundata**
  - An object of class 'optimbase.functionargs' and with (at least) a fun element, which contains the user-defined cost function.

Value

Returns a list with the following elements:

- **f** The value of the cost function at the current point estimate.
index  The same index variable.
this  A list with a single element costargument which contains fmsfundata.

Author(s)
Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also
fminsearch, neldermead, neldermead.set.

fminsearch.outputfun  fminsearch Output Function Call

Description
This function calls the output function and make it match neldermead requirements. It is used in the fminsearch function as the outputcommand element of the neldermead object (see ?neldermead and ?neldermead.set).

Usage
fminsearch.outputfun(state = NULL, data = NULL, fmsdata = NULL)

Arguments
state  The current state of the algorithm either 'init', 'iter' or 'done'.
data  The data at the current state. This is an object of class 'neldermead.data', i.e. a list with the following elements:
   x  The current parameter estimates.
   fval  The current value of the cost function.
   simplex  The current simplex object.
   iteration  The number of iterations performed.
   funccount  The number of function evaluations.
   step  The type of step in the previous iteration.
fmsdata  This is an object of class 'optimbase.functionargs' which contains specific data of the fminsearch algorithm:
   Display  what to display
   OutputFcn  the array of output functions
   PlotFcns  the array of plot functions

Value
This function does not return any data, but execute the output function(s).
Author(s)
Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also
fminsearch, neldermead, neldermead.set.

neldermead S3 neldermead object

Description
These functions support the S3 class 'neldermead' and are intended to either create objects of this class or check if an object is of this class.

Usage
neldermead(optbase, method, simplex0, simplex0method,
  simplex0length, simplex0size0, simplex0opt, historysimplex, coords0, rho, chi,
  gamma, sigma, tolfstdeviation, tolfstdeviationmethod, tolsimplexizeabsolute,
  tolsimplexizerelative, tolsimplexizemethod, toldeltafv, tolsizedeltafvmethod,
  simplex0deltausual, simplex0deltazer0, restartsimplexmethod, restartmax,
  restarteps, restartstep, restartn0b, restartflag, restartdetection,
  kelleystagnationflag, kelleynormalizationflag, kelleystagnationalpha0,
  kelleyalpha, startupflag, boxnbpoints, boxnbpointseff, boxineqscaling,
  checkcostfunction, scalingsimplex0, guinalphamin, boxboundsalpha,
  boxtermination, boxtolf, boxnbmatch, boxkount, boxreflect, tolvarianceflag,
  tolaboluselvariance, tolrelativevariance, variancesimplex0, mymethod,
  myterminate, myterminateflag, greedy, output, exitflag)

  ## S3 method for class 'neldermead'
  print(x,verbose,...)

  ## S3 method for class 'neldermead'
  summary(object,showhistory,...)

  ## S3 method for class 'neldermead'
  is(x=NULL)

Arguments

optbase An object of class 'optimbase', i.e. a list created by optimbase() and containing the following elements:
verbose The verbose option, controlling the amount of messages.
The initial guess.
fx0  The value of the function for the initial guess.
\( \text{xopt} \) The optimum parameter.
\( \text{fopt} \) The optimum function value.
\( \text{tolfunabsolute} \) The absolute tolerance on function value.
\( \text{tolfunrelative} \) The relative tolerance on function value.
\( \text{tolfunmethod} \) Logical flag for the tolerance on function value in the termination criteria. This criteria is suitable for functions which minimum is associated with a function value equal to 0.
\( \text{tolxabsolute} \) The absolute tolerance on x.
\( \text{tolxrelative} \) The relative tolerance on x.
\( \text{tolxmethod} \) Possible values: FALSE, TRUE.
\( \text{funevals} \) The number of function evaluations.
\( \text{maxfunevals} \) The maximum number of function evaluations.
\( \text{maxiter} \) The maximum number of iterations.
\( \text{iterations} \) The number of iterations.
\( \text{fun} \) The cost function.
\( \text{status} \) The status of the optimization.
\( \text{historyfopt} \) The vector to store the history for fopt. The values of the cost function will be stored at each iteration in a new element, so the length of \( \text{historyfopt} \) at the end of the optimization should be the number of iterations.
\( \text{historyxopt} \) The list to store the history for xopt. The vectors of estimates will be stored on separated levels of the list, so the length of \( \text{historyfopt} \) at the end of the optimization should be the number of iterations.
\( \text{verbosetermination} \) The verbose option for termination criteria.
\( \text{outputcommand} \) The command called back for output.
\( \text{outputcommandarg} \) The outputcommand argument is initialized as a string. If the user configure this element, it is expected that a matrix of values or a list is passed so that the argument is appended to the name of the function.
\( \text{numberofvariables} \) The number of variables to optimize.
\( \text{storehistory} \) The flag which enables/disables the storing of the history.
\( \text{costfargument} \) The costf argument is initialized as a string. If the user configure this element, it is expected that a matrix of values or a list is passed so that the argument is appended to the name of the function.
\( \text{boundsmin} \) Minimum bounds for the parameters.
\( \text{boundsmax} \) Maximum bounds for the parameters.
\( \text{nbineqconst} \) The number of nonlinear inequality constraints.
\( \text{logfile} \) The name of the log file.
\( \text{logfilehandle} \) The handle for the log file.
\( \text{logstartup} \) Set to TRUE when the logging is started up.
\( \text{withderivatives} \) Set to TRUE when the method uses derivatives.
\( \text{method} \) The name of the algorithm to use.
simplex0 An object of class 'simplex', i.e. a list created by optimsimplex(), and containing the following elements:

- **verbose** The verbose option, controlling the amount of messages.
- **x** The coordinates of the vertices, with size nbve x n.
- **n** The dimension of the space.
- **fv** The function values, with size nbve x 1.
- **nbve** The number of vertices.

**simplex0method** The method to use to compute the initial simplex.

**simplex0length** The length to use when the initial simplex is computed with the 'axes' or 'spendley' methods.

- **rho** The reflection coefficient. This parameter is used when the method element is set to 'fixed' or 'variable'.

- **chi** The expansion coefficient. This parameter is used when the method element is set to 'variable'.

- **gamma** The contraction coefficient. This parameter is used when the method element is set to 'variable'.

- **sigma** The shrinkage coefficient. This parameter is used when the method element is set to 'fixed' or 'variable'.

**tolfstddeviation** The tolerance for the standard deviation.

**tolfstddeviationmethod** Set to FALSE.

**tolsimplexizeabsolute** The absolute tolerance on the simplex size.

**tolsimplexizerelative** The relative tolerance on the simplex size.

**tolsimplexizemethod** Logical flag to enable/disable the tolerance on the simplex size. When this criteria is enabled, the values of the tolsimplexizeabsolute and tolsimplexizerelative elements are used in the termination criteria. The method to compute the size is the 'sigmaplus' method.

**simplexsize0** Initial size of the simplex, for the tolerance on the simplex size.

**toldeltafv** The absolute tolerance on the difference between the highest and the lowest function values.

**tolssizedeltafvmethod** Logical flag to enable/disable the termination criteria based on the size of the simplex and the difference of function value in the simplex. If this criteria is triggered, the status of the optimization is set to 'tolssizedeltafv'. This termination criteria uses the values of the tolsimplexizeabsolute and toldeltafv elements. This criteria is identical to Scilab’s fminsearch.

**historysimplex** The list to store the history for simplex. The simplex will be stored on a new level of the list at each iteration, so the length of historiesopt at the end of the optimization should be the number of iterations.
coords

The coordinates of the vertices of the initial simplex. If the `simplex0method` element is set to 'given', these coordinates are used to compute the initial simplex. This matrix is expected to have shape nbve x n where nbve is the number of vertices and n is the number of variables.

`simplex0deltausual`

The relative delta for non-zero parameters in 'pfeffer' method.

`simplex0deltazero`

The absolute delta for non-zero parameters in 'pfeffer' method.

`simplexopt`

The optimum simplex, after one optimization process.

`restartsimplexmethod`

The method to compute the initial simplex after a restart.

`restartmax`

The maximum number of restarts, when automatic restart is enabled via the `restartflag` element.

`restarteps`

The absolute epsilon value used to check for optimality in the factorial O’Neill restart detection.

`restartstep`

The absolute step length used to check for optimality in the factorial O’Neill restart detection.

`kelleystagnationflag`

Logical flag to enable/disable the termination criteria using Kelley’s stagnation detection, based on sufficient decrease condition. If this criteria is triggered, the status of the optimization is set to 'kelleystagnation'.

`kelleynormalizationflag`

Logical flag to enable/disable the normalization of the alpha coefficient in Kelley’s stagnation detection, i.e. use the value of the `kelleystagnationalpha` element as is.

`kelleystagnationalpha`

The parameter used in Kelley’s stagnation detection.

`kelleyalpha`

The current value of Kelley’s alpha, after normalization, if required.

`restartnb`

Number of restarts performed.

`restartflag`

Logical flag to enable/disable the automatic restart of the algorithm.

`restartdetection`

The method to detect if the automatic restart must be performed.

`startupflag`

Set to TRUE when the startup has been performed.

`boxnbpoints`

The number of points in the initial simplex, when the `simplex0method` is set to 'randbounds'. The value of this element is also used to update the simplex when a restart is performed and the `restartsimplexmethod` element is set to 'randbounds'. The default value is so that the number of points is twice the number of variables of the problem.

`boxnbpointseff`

The effective number of points required in the simplex for Box’s algorithm.

`boxineqscaling`

The scaling coefficient used to scale the trial point for function improvement or into the constraints of Box’s algorithm.

`checkcostfunction`

Logical flag to enable/disable the checking of the connection of the cost function.
The algorithm used to scale the initial simplex into the nonlinear constraints. The following two algorithms are provided:

- `'tox0'` scales the vertices toward the initial guess.
- `'tocentroid'` scales the vertices toward the centroid, as recommended by Box.

If the centroid happens to be unfeasible, because the constraints are not convex, the scaling of the initial simplex toward the centroid may fail. Since the initial guess is always feasible, scaling toward the initial guess cannot fail.

The minimum value of alpha when scaling the vertices of the simplex into nonlinear constraints in Box's algorithm.

The parameter used to project the vertices into the bounds in Box's algorithm.

Logical flag to enable/disable Box's termination criteria.

The absolute tolerance on difference of function values in the simplex, suggested by Box. This tolerance is used if the `boxtermination` element is set to TRUE.

The number of consecutive match of Box's termination criteria.

Current number of consecutive match.

The reflection factor in Box's algorithm.

Logical flag to enable/disable the termination criteria based on the variance of the function value. If this criteria is triggered, the status of the optimization is set to 'tolvariance'. This criteria is suggested by Nelder and Mead.

The absolute tolerance on the variance of the function values of the simplex.

The relative tolerance on the variance of the function values of the simplex.

Relative tolerance on variance.

A user-defined simplex algorithm.

A user-defined terminate function.

Logical flag to enable/disable the user-defined terminate function.

Logical flag to enable/disable greedy Nelder-Mead.

The command to call back for user-defined output of specialized function.

Logical flag to enable/disable the user-defined output of specialized function.

An object of class 'neldermead'.

A logical flag, controlling the amount of data printed.

Optional arguments to 'print' or 'plot' methods.

An object of class 'neldermead'.

Optional logical flag, to define whether optimization history must be summarized or not.
Value

The `neldermead` function returns a new object of class `’neldermead’`, with the following default content:

**optbase**  An object of class `’optimbase’` with the following default content:

- **verbose**  Default is `FALSE`.
- **x0**  Default is `NULL`.
- **fx0**  Default is `NULL`.
- **xopt**  Default is `0`.
- **fopt**  Default is `0`.
- **tolfunabsolute**  Default is `0`.
- **tolfunrelative**  Default is `Machine$double.eps`.
- **tolfunmethod**  Default is `FALSE`.
- **tolxabsolute**  Default is `0`.
- **tolxrelative**  Default is `Machine$double.eps`.
- **tolxmethod**  Default is `TRUE`.
- **funevals**  Default is `0`.
- **maxfunevals**  Default is `100`.
- **maxiter**  Default is `100`.
- **iterations**  Default is `0`.
- **fun**  Default is `”`.
- **status**  Default is `”`.
- **historyfopt**  Default is `NULL`.
- **historyxopt**  Default is `NULL`.
- **verbosetermination**  Default is `FALSE`.
- **outputcommand**  Default is `”`.
- **outputcommandarg**  Default is `”`. If the user configures this element, it is expected to be an object of class `’optimbase.outputargs’` or will be coerced to an object of class `’optimbase.outputargs’`.
- **numberofvariables**  Default is `0`.
- **storehistory**  Default is `FALSE`.
- **costfargument**  Default is `”`. If the user configures this element, it is expected to be an object of class `’optimbase.functionargs’` or will be coerced to an object of class `’optimbase.functionargs’`.
- **boundsmin**  Default is `NULL`.
- **boundmax**  Default is `NULL`.
- **nbineqconst**  Default is `0`.
- **logfile**  Default is `”`.
- **logfilehandle**  Default is `0`.
- **logstartup**  Default is `FALSE`.
- **withderivatives**  Default is `FALSE`.
- **method**  Default is `’variable’`.
- **simplex0**  Default is an object of class `’simplex’`, with the following content:
verbose Default is 0.
x Default is NULL.
n Default is 0.
fv Default is NULL.
nbve Default is 0.
simplex0method Default is 'axes'.
simplex0length Default is 1.
rho Default is 1.
chi Default is 2.
gamma Default is 0.5.
sigma Default is 0.5.
tolfstdeviation Default is 0.
tolfstdeviationmethod Default is FALSE.
tolsimplexizeabsolute Default is 0.
tolsimplexizerelative Default is .Machine$double.eps.
tolsimplexizemethod Default is FALSE.
simplexsize0 Default is 0.
toldeltafv Default is .Machine$double.eps.
tolssizedeltafvmethod Default is FALSE.
historysimplex Default is NULL.
coords0 Default is NULL.
simplex0deltausual Default is 0.05.
simplex0deltazero Default is 0.0075.
simplexopt Default is NULL.
restartsimplexmethod Default is 'oriented'.
restartmax Default is 3.
restarteps Default is .Machine$double.eps.
restartstep Default is 1.
kelleystagnationflag Default is FALSE.
kelleynormalizationflag Default is TRUE, i.e. the simplex gradient of the initial simplex is taken into account in the stagnation detection.
kelleystagnationalpha0 Default is 1.e-4.
kellaleyalpha Default is 1.e-4.
restartnb Default is 0.
restartflag Default is FALSE.
restartdetection Default is 'oneill'.
startupflag Default is FALSE.
boxnbpoints Default is '2n'.

boxnbpointseff Default is 0.
boxineqscaling Default is 0.
checkcostfunction Default is TRUE.
scalingsimplex0 Default is 'tox0'.
guinalphamin Default is 1.e-6.
boxtermination Default is FALSE.
boxtolf Default is 1.e-5.
boxnbmatch Default is 5.
boxkount Default is 0.
boxreflect Default is 1.3.
tolvarianceflag Default is FALSE.
tolabsolutevariance Default is 0.
tolrelativevariance Default is .Machine$double.eps.
variancesimplex0 Default is .Machine$double.eps.
mymethod Default is NULL.
myterminate Default is NULL.
myterminateflag Default is FALSE.
greedy Default is FALSE.
output Default is list().
exitflag Default is FALSE.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

optimbase, optimsimplex

---

**neldermead.algo**  
*Nelder-Mead Algorithm*

**Description**

neldermead.algo performs an optimization without restart using the method associated with the method element of the neldermead object; neldermead.fixed, neldermead.variable, neldermead.box, boxlinesearch, neldermead.storehistory, neldermead.termination, and neldermead.interpolate are utility functions for neldermead.algo.
Usage

neldermead.algo(this = NULL)
neldermead.fixed(this = NULL)
neldermead.variable(this = NULL)
neldermead.box(this = this)
boxlinesearch(this = NULL, n = NULL, xbar = NULL, xhigh = NULL, fhigh = NULL, 
rho = NULL)
neldermead.storehistory(this = NULL, n = NULL, fopt = NULL, xopt = NULL, 
fv = NULL, xcoords = NULL)
neldermead.termination(this = NULL, fvinitial = NULL, oldfvmean = NULL, 
newfvmean = NULL, previousxopt = NULL, 
currentxopt = NULL, simplex = NULL)
neldermead.interpolate(x1 = NULL, x2 = NULL, fac = NULL)

Arguments

this A neldermead object.
n Number of variables.

xbar The centroid.

xhigh The high point.

fhigh The value of the cost function at xhigh.

rho The reflection factor.

fopt The current value of the function at the current optimum point estimate.

xopt The current optimum point estimate.

fv The function values, with size nbve x 1.

xcoords Matrix of size n x n+1, coordinates of the n+1 vertices

fvinitial The initial cost function value.

oldfvmean The old cost function value average on the simplex.

newfvmean The new cost function value average on the simplex.

previousxopt The previous point estimate.

currentxopt The current point estimate.

simplex The simplex. The best point estimate in the simplex is expected to be stored at 
1, while the worst point estimate in the simplex is expected to be stored at n+1.

x1 The first reference point estimate to perform the interpolation.

x2 The second reference point estimate to perform the interpolation.

fac A factor to perform the interpolation.

Details

neldermead.fixed The simplex algorithm with fixed size simplex. We implement the following 'rules' of the method of Spendley et al.

• Rule 1 is strictly applied, but the reflection is done by reflection of the high point, since we minimize a function instead of maximizing it, like Spendley.
• Rule 2 is NOT implemented, as we expect that the function evaluation is not subject to errors.
• Rule 3 is applied, i.e. reflection with respect to next to high point. A shrink step is included, with shrinkage factor sigma.

Rule 1. Ascertain the lowest reading \( y \), of \( y_i \ldots y_{k+1} \) Complete a new simplex \( S_p \) by excluding the point \( V_p \) corresponding to \( y \), and replacing it by \( V^* \) defined as above.

Rule 2. If a result has occurred in \( (k + 1) \) successive simplexes, and is not then eliminated by application of Rule 1, do not move in the direction indicated by Rule 1, or at all, but discard the result and replace it by a new observation at the same point.

Rule 3. If \( y \) is the lowest reading in \( S_o \), and if the next observation made, \( y^* \), is the lowest reading in the new simplex \( S \), do not apply Rule 1 and return to \( S_o \) from \( S_p \). Move out of \( S \), by rejecting the second lowest reading (which is also the second lowest reading in \( S_o \)).

\textbf{neldermead.variable} The original Nelder-Mead algorithm, with variable-size simplex.

\textbf{neldermead.box} The Nelder-Mead algorithm, with variable-size simplex and modifications by Box for bounds and inequality constraints.

\textbf{boxlinesearch} Called by \textbf{neldermead.box}, i.e. Box’s method. Perform a line search from \( x_{bar} \), on the line \((x_{high},x_{bar})\). The reflected point estimate satisfies the following constraints:

- \( f_r < f_{high} \)
- \( x_r \) satisfies the bounds constraints
- \( x_r \) satisfies the nonlinear positive inequality constraints
- \( x_r \) satisfies the linear positive inequality constraints

The method is based on projection and scaling toward the centroid.

\textbf{neldermead.storehistory} Store the optimization history into the \textbf{neldermead} object.

\textbf{neldermead.termination} Determine if the algorithm must continue or terminate. The function uses the cost function average in the simplex instead of the best cost function value. This is because the function average changes at each iteration. Instead, the best function value has a step-by-step evolution and may not change between two successive iterations, leading to a stop of the algorithm.

\textbf{neldermead.interpolate} Compute the point estimate \( x_i \) as an interpolation between \( x_1 \) and \( x_2 \), as follows: \( x_i = (1 + fac)x_1 - fac*x_2 \)

\textbf{Value}

\textbf{neldermead.fixed}, \textbf{neldermead.variable}, and \textbf{neldermead.box} Return the updated \textbf{neldermead} object, containing the optimum point estimate.

\textbf{boxlinesearch} Return a list with the following elements:

- \textbf{this} The updated \textbf{neldermead} object.
- \textbf{status} TRUE if the search is successful, FALSE otherwise.
- \textbf{xr} The reflected point estimate.
- \textbf{fr} The value of the cost function at \( x_r \).

\textbf{neldermead.storehistory} Return the updated \textbf{neldermead} object.

\textbf{neldermead.termination} Return a list with the following elements:

- \textbf{this} The updated \textbf{neldermead} object
terminate  TRUE if the algorithm terminates, FALSE if the algorithm must continue.
status  The termination status: 'continue', 'maxiter', 'maxfuneval', 'tolf', 'tolx', 'tolsize',
        'tolsizedeltafv', 'kelleystagnation', 'tolboxf', 'tolvariance' or the user-defined termina-
        tion status.

neldermead.interpólate  Return a new point estimate, i.e. a column vector.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

neldermead.destroy  Erase a neldermead object.

Description

neldermead.destroy calls optimbase.destroy and optimsimplex.destroy to erase the content
of this$optbase and this$simplex$.

Usage

neldermead.destroy(this = NULL)

Arguments

this  A neldermead object.

Value

Return an updated neldermead object.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

optimbase.destroy, optimsimplex.destroy
neldermead.function  

*Call Cost Function.*

**Description**

Simple way to compute the value of the cost function specified in a neldermead object.

**Usage**

neldermead.function(this = NULL, x = NULL)

**Arguments**

- **this**: A neldermead object.
- **x**: The point estimate where the cost function is to be evaluated.

**Value**

Returns the value of the cost function.

**Author(s)**

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

neldermead.get  

*Get the value for the given element*

**Description**

Get the value for the given element in a neldermead object.

**Usage**

neldermead.get(this = NULL, key = NULL)

**Arguments**

- **this**: A neldermead object.
- **key**: The name of the key to query.

**Value**

Return the value of the list element key, or an error message if key does not exist in the neldermead object this.
neldermead.restart

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

neldermead.set, optimbase.get

Description

Update the simplex with neldermead.updatesimp and restart the search with neldermead.search.

Usage

neldermead.restart(this = NULL)

Arguments

this A neldermead object.

Value

Returns an updated neldermead object.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

neldermead.updatesimp, neldermead.search,
neldermead.search  

*Starts the optimization*

**Description**

Performs the optimization associated with the method associated with the method element of the neldermead object and find the optimum. If the restartflag element is enabled, automatic restarts are performed, based on the restartdetection element.

**Usage**

```r
neldermead.search(this = NULL)
```

**Arguments**

- **this**: A neldermead object.

**Value**

Return an updated neldermead object.

**Author(s)**

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)

Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

**See Also**

fminsearch, neldermead, neldermead.set,

neldermead.set  

*Neldermead Object Configuration*

**Description**

Configure the current neldermead object with the given value for the given key.

**Usage**

```r
neldermead.set(this = NULL, key = NULL, value = NULL)
```

**Arguments**

- **this**: The current neldermead object.
- **key**: The key to configure. See details for the list of possible keys.
- **value**: The value to assign to the key.
Details

neldermead.set sets the content of the key element of the neldermead object this to value. If key is a sub-element of this$optbase, value is assigned by optimbase.set.

The main available keys are the following:

'\-verbose' Set to 1 to enable verbose logging.

'\-verbosetermination' Set to 1 to enable verbose termination logging.

'\-x0' The initial guess, as a n x 1 column vector, where n is the number of variables.

'\-maxfunevals' The maximum number of function evaluations. If this criteria is triggered during optimization, the status of the optimization is set to 'maxfuneval'.

'\-maxiter' The maximum number of iterations. If this criteria is triggered during optimization, the status of the optimization is set to 'maxiter'.option

'\-tolfunabsolute' The absolute tolerance for the function value.

'\-tolfunrelative' The relative tolerance for the function value.

'\-tolfunmethod' The method used for the tolerance on function value in the termination criteria. The following values are available: TRUE, FALSE. If this criteria is triggered during optimization, the status of the optimization is set to 'tolf'.

'\-tolxabsolute' The absolute tolerance on x.

'\-tolxrelative' The relative tolerance on x.

'\-tolxmethod' The method used for the tolerance on x in the termination criteria. The following values are available: TRUE, FALSE. If this criteria is triggered during optimization, the status of the optimization is set to 'tolx'.

'\-function' The objective function, which computes the value of the cost and the non linear constraints, if any. See vignette('neldermead',package='neldermead') for the details of the communication between the optimization system and the cost function.

'\-costfargument' An additional argument, passed to the cost function.

'\-outputcommand' A command which is called back for output. See vignette('neldermead',package='neldermead') for the details of the communication between the optimization system and the output command function.

'\-outputcommandarg' An additional argument, passed to the output command.option

'\-numberofvariables' The number of variables to optimize.

'\-storehistory' Set to TRUE to enable the history storing.

'\-boundsmin' The minimum bounds for the parameters.

'\-boundsmax' The maximum bounds for the parameters.

'\-nbineqconst' The number of inequality constraints.

'\-method' The name of the algorithm to use. The following methods are available:

'\fixed' the fixed simplex shape algorithm of Spendley et al. This algorithm is for unconstrained problems (i.e. bounds and non linear constraints are not taken into account)

'\variable' the variable simplex shape algorithm of Nelder and Mead. This algorithm is for unconstrained problems (i.e. bounds and non linear constraints are not taken into account)

'\box' Box's complex algorithm. This algorithm takes into account bounds and nonlinear inequality constraints.
'mine' the user-defined algorithm, associated with the mymethod element. See vignette('neldermead',package='neldermead') for details.

'-simplex0method' The method to use to compute the initial simplex. The first vertex in the simplex is always the initial guess associated with the x0 element. The following methods are available:

'given' The coordinates associated with the coords0 element are used to compute the initial simplex, with arbitrary number of vertices. This allows the user to setup the initial simplex by a specific method which is not provided by the current package (for example with a simplex computed from a design of experiments). This allows also to configure the initial simplex so that a specific behaviour of the algorithm is to be reproduced (for example the Mac Kinnon test case). The given matrix is expected to have nbve rows and n columns, where n is the dimension of the problem and nbve is the number of vertices.

'axes' The simplex is computed from the coordinate axes and the length associated with the simplex0length element.

'spendley' The simplex is computed so that it is regular with the length associated with the simplex0length element (i.e. all the edges have the same length).

'pfeffer' The simplex is computed from an heuristic, in the neighborhood of the initial guess. This initial simplex depends on the simplex0deltausual and simplex0deltazero.

'randbounds' The simplex is computed from the bounds and a random number. This option is available only if bounds are available: if bounds are not available, an error is generated. This method is usually associated with Box’s algorithm. The number of vertices in the simplex is taken from the boxnbpoints element.

'-coords0' The coordinates of the vertices of the initial simplex. If the simplex0method element is set to 'given', these coordinates are used to compute the initial simplex. This matrix is expected to have shape nbve x n, where nbve is the number of vertices and n is the number of variables.

'-simplex0length' The length to use when the initial simplex is computed with the 'axes' or 'spendley' methods. If the initial simplex is computed from 'spendley' method, the length is expected to be a scalar value. If the initial simplex is computed from 'axes' method, it may be either a scalar value or a vector of values, of length n, where n is the number of variables.

'-simplex0deltausual' The relative delta for non-zero parameters in 'pfeffer' method.

'-simplex0deltazero' The absolute delta for non-zero parameters in 'pfeffer' method.

'-rho' The reflection coefficient. This parameter is used when the method element is set to 'fixed' or 'variable'.

'-chi' The expansion coefficient. This parameter is used when the method element is set to 'variable'.

'-gamma' The contraction coefficient. This parameter is used when the method element is set to 'variable'.

'-sigma' The shrinkage coefficient. This parameter is used when the method element is set to 'fixed' or 'variable'.

'-tolsimplexizemethod' Set to FALSE to disable the tolerance on the simplex size. If this criteria is triggered, the status of the optimization is set to 'tolsize'. When this criteria is enabled, the values of the tolsimplexizeabsolute and tolsimplexizerelative elements are used in the termination criteria. The method to compute the size is the 'sigmaplus' method.
'-tolsimplexizeabsolute' The absolute tolerance on the simplex size.
'-tolsimplexizerelative' The relative tolerance on the simplex size.
'-tolsizedeltafvmethod' Set to TRUE to enable the termination criteria based on the size of the simplex and the difference of function value in the simplex. If this criteria is triggered, the status of the optimization is set to 'tolsizedeltafv'. This termination criteria uses the values of the tolsimplexizeabsolute and toldeltafv elements.option
'-toldeltafv' The absolute tolerance on the difference between the highest and the lowest function values.
'-tolvarianceflag' Set to TRUE to enable the termination criteria based on the variance of the function value. If this criteria is triggered, the status of the optimization is set to 'tolvariance'. This criteria is suggested by Nelder and Mead.
'-tolabsolutevariance' The absolute tolerance on the variance of the function values of the simplex.
'-tolrelativevariance' The relative tolerance on the variance of the function values of the simplex.
'-kelleystagnationflag' Set to TRUE to enable the termination criteria using Kelley’s stagnation detection, based on sufficient decrease condition. If this criteria is triggered, the status of the optimization is set to 'kelleystagnation'.
'-kelleynormalizationflag' Set to FALSE to disable the normalization of the alpha coefficient in Kelley’s stagnation detection, i.e. use the value of the kelleystagnationalpha0 element as is. Default value is TRUE, i.e. the simplex gradient of the initial simplex is takeoptionn into account in the stagnation detection.
'-kelleystagnationalpha0' The parameter used in Kelley’s stagnation detection.
'-restartflag' Set to TRUE to enable the automatic restart of the algorithm.
'-restartdetection' The method to detect if the automatic restart must be performed. The following methods are available:
  'oneill' The factorial local optimality test by O’Neill is used. If the test finds a local point which is better than the computed optimum, a restart is performed.
  'kelley' The sufficient decrease condition by O’Neill is used. If the test finds that the status of the optimization is ‘kelleystagnation’, a restart is performed. This status may be generated if the -kelleystagnationflag option is set to TRUE.
'-restartmax' The maximum number of restarts, when automatic restart is enabled via the -restartflag option.
'-restarteps' The absolute epsilon value used to check for optimality in the factorial O’Neill restart detection.
'-restartstep' The absolute step length used to check for optimality in the factorial O’Neill restart detection.
'-restartsimplexmethod' The method to compute the initial simplex after a restart. The following methods are available.
  'given' The coordinates associated with the coords0 element are used to compute the initial simplex, with arbitrary number of vertices. This allow the user to setup the initial simplex by a specific method which is not provided by the current package (for example with a simplex computed from a design of experiments). This allows also to configure the initial simplex so that a specific behaviour of the algorithm is to be reproduced (for example the Mc Kinnon test case). The given matrix is expected to have nbve rows and n columns, where n is the dimension of the problem and nbve is the number of vertices.
'axes' The simplex is computed from the coordinate axes and the length associated with the -simplex0length option.

'spendley' The simplex is computed so that it is regular with the length associated with the -simplex0length option (i.e. all the edges have the same length).

'pfeffer' The simplex is computed from a heuristic, in the neighborhood of the initial guess. This initial simplex depends on the -simplex0deltausual and -simplex0deltazero.

'randbounds' The simplex is computed from the bounds and a random number. This option is available only if bounds are available: if bounds are not available, an error is generated. This method is usually associated with Box’s algorithm. The number of vertices in the simplex is taken from the -boxnbpoints option.

'oriented' The simplex is computed so that it is oriented, as suggested by Kelley.

'scalingsimplex0' The algorithm used to scale the initial simplex into the nonlinear constraints. The following two algorithms are provided:

'tox0' scales the vertices toward the initial guess.
'tocentroid' scales the vertices toward the centroid, as recommended by Box.

If the centroid happens to be unfeasible, because the constraints are not convex, the scaling of the initial simplex toward the centroid may fail. Since the initial guess is always feasible, scaling toward the initial guess cannot fail.

'boxnbpoints' The number of points in the initial simplex, when the -simplex0method is set to 'randbounds'. The value of this option is also use to update the simplex when a restart is performed and the -restartsimplexmethod option is set to 'randbounds'. The default value is so that the number of points is twice the number of variables of the problem.

'boxineqscaling' The scaling coefficient used to scale the trial point for function improvement or into the constraints of Box’s algorithm.

'guinalphamin' The minimum value of alpha when scaling the vertices of the simplex into nonlinear constraints in Box’s algorithm.

'boxreflect' The reflection factor in Box’s algorithm.

'boxtermination' Set to TRUE to enable Box’s termination criteria.

'boxtolf' The absolute tolerance on difference of function values in the simplex, suggested by Box. This tolerance is used if the -boxtermination element is set to TRUE.

'boxnbmatch' The number of consecutive match of Box’s termination criteria.

'boxboundsalpha' The parameter used to project the vertices into the bounds in Box’s algorithm.

'mymethod' A user-defined simplex algorithm. See vignette('neldermead',package='neldermead') for details.

'myterminate' A user-defined terminate function. See vignette('neldermead',package='neldermead') for details.

'myterminateflag' Set to TRUE to enable the user-defined terminate function.

Value

An updated neldermead object.
Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

neldermead

Description

This function allows to make queries on an existing optimization option list. This list must have been created and updated by the optimset function. The optimget allows to retrieve the value associated with a given key.

Usage

optimget(options = NULL, key = NULL, value = NULL)

Arguments

options A list created or modifies by optimset.
key A single character string, which should be the name of the field in options to query (case insensitive).
value A default value.

Details

key is matched against the field names of options using grep and a case-insensitive regular expression. If key is not found in options, the function returns NULL. If several matches are found, optimget is stopped.

Value

Return options$\texttt{key}$ if key is found in options. Return value, otherwise.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

optimset
optimset

Configures and returns an optimization data structure.

Description

This function creates or updates a list which can be used to modify the behaviour of optimization methods. The goal of this function is to manage the options list with a set of fields (for example, 'MaxFunEvals', 'MaxIter', etc...). The user can create a new list with empty fields or create a new structure with default fields which correspond to a particular algorithm. The user can also configure each field and set it to a particular value. Finally, the user passes the list to an optimization function so that the algorithm uses the options configured by the user.

Usage

optimset(method = NULL,...)

Arguments

- method: If provided, the method calls the optimset.method function. If the content of method is recognized, a default set of options are returned. The only current recognized character strings are 'fminsearch' and 'fminbnd'.

- ...: Additional arguments which would be included in the options output if the method argument is not used. See Details.

Details

Most optimization algorithms require many algorithmic parameters such as the number of iterations or the number of function evaluations. If these parameters are given to the optimization function as input parameters, this forces both the user and the developer to manage many input parameters. The goal of the optimset function is to simplify the management of input arguments, by gathering all the parameters into a single list.

While the current implementation of the optimset function only supports the fminsearch and fminbnd function, it is designed to be extended to as many optimization function as required. Because all optimization algorithms do not require the same parameters, the data structure aims at remaining flexible. But, most of the time, most parameters are the same from algorithm to algorithm, for example, the tolerance parameters which drive the termination criteria are often the same, even if the termination criteria itself is not the same.

Optimization parameters that are returned by the optimset function and that can be defined in ... are the following:

- **Display**: The verbose level. The default value is 'notify'. The following is a list of available verbose levels.
'off'  The algorithm displays no message at all.

'notify'  The algorithm displays message if the termination criteria is not reached at the end of
          the optimization. This may happen if the maximum number or iterations of the maximum number of function evaluations is reached and warns the user of a convergence problem.

'final'  The algorithm displays a message at the end of the optimization, showing the number of iterations, the number of function evaluations and the status of the optimization. This option includes the messages generated by the 'notify' option i.e. warns in case of a convergence problem.

'iter'  The algorithm displays a one-line message at each iteration. This option includes the messages generated by the 'notify' option i.e. warns in case of a convergence problem. It also includes the message generated by the 'final' option.

FunValCheck  A logical flag to enable the checking of function values.

MaxFunEvals  The maximum number of evaluations of the cost function.

MaxIter  The maximum number of iterations.

OutputFcn  A function which is called at each iteration to print out intermediate state of the optimization algorithm (for example into a log file).

PlotFcns  A function which is called at each iteration to plot the intermediate state of the optimization algorithm (for example into a 2D graphic).

TolFun  The absolute tolerance on function value.

TolX  The absolute tolerance on the variable x.

nbMatch  Specific to Box method: the number of consecutive times the TolFun criteria must be met to terminate the optimization.

boundsAlpha  Specific to Box method: the parameter used to project the vertices into the bounds in Box’s algorithm.

boxScaling  Specific to Box method: the scaling coefficient used to scale the trial point for function improvement or into the constraints of Box’s algorithm.

alphaMin  Specific to Box method: the minimum value of alpha when scaling the vertices of the simplex into nonlinear constraints in Box’s algorithm.

Output and plot functions  The 'OutputFcn' and 'PlotFcns' options accept as argument a function (or a list of functions). In the client optimization algorithm, this output or plot function is called back once per iteration. It can be used by the user to display a message in the console, write into a file, etc... The output or plot function is expected to have the following definition:

myfun <- function(x, optimValues, state)

where the input parameters are:

x  The current point estimate.

optimValues  A list which contains the following fields:

  funcount  The number of function evaluations.
  fval  The best function value.
  iteration  The current iteration number.
  procedure  The type of step performed. This string depends on the specific algorithm (see fminsearch for details).
state  the state of the algorithm. The following states are available:

- 'init' when the algorithm is initializing,
- 'iter' when the algorithm is performing iterations,
- 'done' when the algorithm is terminated.

Value

Return a list with the following fields: Display, FunValCheck, MaxFunEvals, MaxIter, OutputFcn, PlotFcns, TolFun, TolX, nbMatch, boundsAlpha, boxScaling, and alphaMin.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

optimset.method, fminsearch, fminbnd

Examples

optimset()
optimset(Display='iter')
optimset(method='fminbnd')

---

optimset.method  Default set of optimization options

Description

This function returns a default set of optimization options for defined 'methods': optimset.method is called by optimset when a method was provided as input. Currently, the only valid method is 'fminsearch'.

Usage

optimset.method(method = NULL)

Arguments

method  A character string.

Value

Returns a list with the following fields: Display, FunValCheck, MaxFunEvals, MaxIter, OutputFcn, PlotFcns, TolFun, and TolX.
Secondary search functions

Author(s)
Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also
optimset

Examples

optimset.method('fminsearch')
  ## Not run: optimset.method('abc')

---

Secondary search functions

Secondary functions for neldermead.search

Description

Utility functions for neldermead.start and dependent functions.

Usage

neldermead.startup(this = NULL)
neldermead.log(this = NULL, msg = NULL)
neldermead.scaletox0(this = NULL, simplex0 = NULL)
neldermead.scaletoocenter(this = NULL, simplex0 = NULL, x0 = NULL)
neldermead.termdstartup(this = NULL)
neldermead.outputcmd(this = NULL, state = NULL, simplex = NULL, step = NULL)
neldermead.autorestart(this = NULL)
neldermead.istorestart(this = NULL)
neldermead.isroneill(this = NULL)
neldermead.isrkelley(this = this)
neldermead.updateesimp(this = NULL)
scaleinstraints(this = NULL, x = NULL, xref = NULL)
neldermead.costf(x = NULL, this = NULL)

Arguments

this A neldermead object.
msg A character string.
simplex0 The initial simplex object.
x0 A column matrix of initial parameters.
state The state of the algorithm, either ‘init’, ‘done’ or ‘iter’.
simplex The current simplex object.
Secondary search functions

step
The type of step performed during the iteration: 'init', 'done', 'reflection', 'expansion', 'insidecontraction', 'outsidecontraction', 'reflectionnext' or 'shrink'.

x
The point estimate to scale.

xref
The reference point estimate.

Details

neldermead.startup Startup the algorithm. Compute the initial simplex, depending on the content of the simplex.method element of the neldermead object ('given', 'axes', 'spendley', 'pfeffer' or 'randbounds').

neldermead.log Print a message to the log file using optimbase.log.

neldermead.scaletox Scale the simplex into the nonlinear inequality constraints, if any. Scale toward x0, which is feasible.

neldermead.scaletocenter Scale the simplex into the nonlinear inequality constraints, if any. Scale to the centroid of the points which satisfy the constraints. This is Box’s method for scaling. It is unsure, since the centroid of the points which satisfy the constraints may not be feasible.

neldermead.terminstartup Initialize Kelley’s stagnation detection system when normalization is required, by computing kelleyalpha. If the simplex gradient is zero, then use alpha0 as alpha.

neldermead.outputcmd Call the array of user-defined output functions

neldermead.autorestart Perform an optimization with automatic restart. The loop processes for i = 1 to restartmax + 1. This is because a RE-start is performed after one simulation has been performed, hence the 'RE'.

neldermead.istorestart Determine if the optimization is to restart using neldermead.isroneill or neldermead.isrkelley depending on the content of the restartdetection element.

neldermead.isroneill Determine if the optimization is to restart. Use O’Neill method as a criteria for restart. It is an axis-by-axis search for optimality.

neldermead.isrkelley Determine if the optimization is to restart. Use kelleystagnation as a criteria for restart.

neldermead.updatesimp Update the initial simplex simplex0 for a restart.

scaleinconstraints Given a point reference to scale and a reference point which satisfies the constraints, scale the point towards the reference point estimate until it satisfies all the constraints.

neldermead.costf Call the cost function and return the value. This function is given to the simplex function class as a callback. Input/Output arguments are swapped w.r.t. optimbase.function, so that it matches the requirements of simplex methods.

Value

neldermead.startup Return an updated neldermead object this.

neldermead.log Return the neldermead object this.

neldermead.scaletox Return an updated simplex.

neldermead.scaletocenter Return an updated simplex.
Secondary search functions

neldermead.termstartup Return an updated neldermead object this.
neldermead.outputcmd Do not return any data, but execute the output function(s).
neldermead.autorestart Return an updated neldermead object this.
neldermead.istorestart Return a list with the following elements:
  this The input neldermead object.
  istorestart Set to TRUE if the optimization is to restart, to FALSE otherwise.
neldermead.isroneill Return a list with the following elements:
  this The input neldermead object.
  istorestart Set to TRUE if the optimization is to restart, to FALSE otherwise.
neldermead.isrkelley Return a list with the following elements:
  this The input neldermead object.
  istorestart Set to TRUE if the optimization is to restart, to FALSE otherwise.
neldermead.updatesimp Return an updated neldermead object this.
scaleinconstraints Return a list with the following elements:
  this The updated neldermead object.
  isscaled TRUE if the procedure has succeeded before boxnb1loops, FALSE if it has failed.
  p The scaled parameters.
neldermead.costf Return a list with the following elements:
  f The value of the cost function.
  this The updated neldermead object.

Author(s)

Author of Scilab neldermead module: Michael Baudin (INRIA - Digiteo)
Author of R adaptation: Sebastien Bihorel (<sb.pmlab@gmail.com>)
Index

*Topic method
   - costf.transpose, 4
   - fmin.gridsearch, 5
   - fminbnd, 6
   - fminbnd.function, 9
   - fminsearch, 11
   - fminsearch.function, 15
   - fminsearch.outputfun, 16
   - neldermead, 17
   - neldermead.algo, 24
   - neldermead.destroy, 27
   - neldermead.function, 28
   - neldermead.get, 28
   - neldermead.restart, 29
   - neldermead.search, 30
   - neldermead.set, 30
   - optimget, 35
   - optimset, 36
   - optimset.method, 38
   - Secondary search functions, 39

*Topic package
   - neldermead-package, 2

boxlinesearch (neldermead.algo), 24

costf.transpose, 4
fmin.gridsearch, 5
fminbnd, 6, 10, 11, 38
fminbnd.function, 9
fminbnd.outputfun, 10
fminsearch, 6, 11, 16, 17, 30, 38
fminsearch.function, 15
fminsearch.outputfun, 16

is.neldermead (neldermead), 17
neldermead, 10, 11, 14, 16, 17, 17, 30, 35
neldermead-package, 2
neldermead.algo, 24

neldermead.autorestart (Secondary search functions), 39
neldermead.box (neldermead.algo), 24
neldermead.costf, 5
neldermead.costf (Secondary search functions), 39
neldermead.destroy, 27
neldermead.fixed (neldermead.algo), 24
neldermead.function, 28
neldermead.get, 28
neldermead.interpolate (neldermead.algo), 24
neldermead.isrkelley (Secondary search functions), 39
neldermead.isroneill (Secondary search functions), 39
neldermead.istorestart (Secondary search functions), 39
neldermead.log (Secondary search functions), 39
neldermead.outputcmd (Secondary search functions), 39
neldermead.restart, 29
neldermead.scaletocenter (Secondary search functions), 39
neldermead.scaletox0 (Secondary search functions), 39
neldermead.search, 29, 30
neldermead.set, 10, 11, 16, 17, 29, 30, 30
neldermead.startup (Secondary search functions), 39
neldermead.storehistory (neldermead.algo), 24
neldermead.termination (neldermead.algo), 24
neldermead.termstartup (Secondary search functions), 39
neldermead.updatesimp, 29
neldermead.updatesimp (Secondary
INDEX

search functions), 39
neldermead.variable (neldermead.algo), 24

optimbase, 4, 24
optimbase.destroy, 27
optimbase.get, 29
optimbase.gridsearch, 6
optimget, 35
optimset, 9, 14, 35, 36, 39
optimset.method, 38, 38
optim simplex, 4, 24
optim simplex.destroy, 27

print.neldermead (neldermead), 17

scaleinconstraints (Secondary search functions), 39
Secondary search functions, 39
summary.neldermead (neldermead), 17