Package ‘subplex’

April 5, 2018

Version 1.5-4
Date 2018-04-04
Title Unconstrained Optimization using the Subplex Algorithm
License GPL-3
Depends R(>= 2.5.1)
URL https://github.com/kingaa/subplex/

BugReports https://github.com/kingaa/subplex/issues/

Description The subplex algorithm for unconstrained optimization, developed by Tom Rowan <http://www.netlib.org/opt/subplex.tgz>.

NeedsCompilation yes
Author Aaron A. King [aut, trl, cre],
   Tom Rowan [aut]
Maintainer Aaron A. King <kingaa@umich.edu>
Repository CRAN
Date/Publication 2018-04-05 14:33:10 UTC

R topics documented:

subplex-package ........................................................................ 2
subplex .................................................................................... 2

Index 5
The 'subplex' package implements Tom Rowan's subspace-searching simplex algorithm for unconstrained minimization of a function.

Subplex is a subspace-searching simplex method for the unconstrained optimization of general multivariate functions. Like the Nelder-Mead simplex method it generalizes, the subplex method is well suited for optimizing noisy objective functions. The number of function evaluations required for convergence typically increases only linearly with the problem size, so for most applications the subplex method is much more efficient than the simplex method.

Subplex was written in FORTRAN by Tom Rowan (Oak Ridge National Laboratory). The FORTRAN source code is maintained on the netlib repository (netlib.org).

Author(s)
Aaron A. King (kingaa at umich dot edu)

References

See Also
subplex, optim

subplex minimizes a function.

Usage
```
subplex(par, fn, control = list(), hessian = FALSE, ...)
```
Arguments

par  Initial guess of the parameters to be optimized over.

fn  The function to be minimized. Its first argument must be the vector of parameters to be optimized over. It should return a scalar result.

control  A list of control parameters, consisting of some or all of the following:

reltol  The relative optimization tolerance for par. This must be a positive number. The default value is `Machine$double$eps`.

maxit  Maximum number of function evaluations to perform before giving up. The default value is 10000.

parscale  The scale and initial stepsizes for the components of par. This must either be a single scalar, in which case the same scale is used for all parameters, or a vector of length equal to the length of par. For parscale to be valid, it must not be too small relative to par: if $\text{par} + \text{parscale} = \text{par}$ in any component, parscale is too small. The default value is 1.

hessian  If hessian=TRUE, the Hessian of the objective at the estimated optimum will be numerically computed.

...  Additional arguments to be passed to the function fn.

Details

The convergence codes are as follows:

-2  invalid input

-1  number of function evaluations needed exceeds maxnfe

0  success: tolerance tol satisfied

1  limit of machine precision reached

For more details, see the source code.

Value

subplex returns a list containing the following:

par  Estimated parameters that minimize the function.

value  Minimized value of the function.

count  Number of function evaluations required.

convergence  Convergence code (see Details).

message  A character string giving a diagnostic message from the optimizer, or `NULL`.

hessian  Hessian matrix.

Author(s)

Aaron A. King <kingaa@umich.edu>
References


See Also

optim

Examples

rosen <- function (x) {  ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100*(x2-x1*x1)^2+(1-x1)^2
}
subplex(par=c(11,-33),fn=rosen)

rosen2 <- function (x) {
  X <- matrix(x,ncol=2)
  sum(apply(X,1,rosen))
}
subplex(par=c(-33,11,14,9,0,12),fn=rosen2,control=list(maxit=30000))
## compare with optim:
optim(par=c(-33,11,14,9,0,12),fn=rosen2,control=list(maxit=30000))

ripple <- function (x) {
  r <- sqrt(sum(x^2))
  1-exp(-r^2)*cos(10*r)^2
}
subplex(par=c(1),fn=ripple,hessian=TRUE)
subplex(par=c(0.1,3),fn=ripple,hessian=TRUE)
subplex(par=c(0.1,3,2),fn=ripple,hessian=TRUE)

rosen <- function (x, g = 0, h = 0) {  ## Rosenbrock Banana function (using names)
  x1 <- x['a']
  x2 <- x['b']-h
  100*(x2-x1*x1)^2+(1-x1)^2+g
}
subplex(par=c(b=11,a=-33),fn=rosen,h=22,control=list(abstol=1e-9,parscale=5),hessian=TRUE)
Index

*Topic optimize
  subplex, 2
  subplex-package, 2

optim, 2, 4

subplex, 2, 2
subplex-package, 2