

Package ‘nleqslv’

February 19, 2012

Type Package

Title Solve systems of non linear equations

Version 1.9.3

Date 2012-02-22

Author Berend Hasselman

Maintainer Berend Hasselman <bhh@xs4all.nl>

Description Solve a system of non linear equations using a Broyden or a Newton method with a choice of global strategies such as linesearch and trust region. There are options for using a numerical or an analytical jacobian and fixed or automatic scaling of parameters.

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-02-19 17:21:29

R topics documented:

nleqslv-package	2
nleqslv	2
nleqslv-iterationreport	7

Index	10
--------------	-----------

nleqslv-package

Solving Nonlinear Systems of Equations

Description

The **nleqslv** package provides two algorithms for solving (dense) non linear systems of equations. The methods provided are

- a Broyden Secant method where the matrix of derivatives is updated after each major iteration using the Broyden rank 1 update.
- a full Newton method where the Jacobian matrix of derivatives is recalculated at each iteration

Both methods utilise global strategies such as linesearch or trust region methods whenever the standard Newton/Broyden step does not lead to a point closer to a root of the equation system. Linesearch may be either quadratic or geometric. The trust region methods are either the double dogleg or the Powell single dogleg method.

The algorithms provided in this package are derived from Dennis and Schnabel (1996). The code is written in Fortran 77 and Fortran 95 and uses Lapack and BLAS routines as provided by the R system.

Author(s)

Berend Hasselman <bhh@xs4a11.nl>

References

Dennis, J.E. Jr and Schnabel, R.B. (1996), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Siam.

nleqslv

Solving systems of non linear equations with Broyden or Newton

Description

The function solves a system of non linear equations with either a Broyden or a full Newton method. It provides linesearch and trust region global strategies for difficult systems.

Usage

```
nleqslv(x, fn, jac=NULL, ...,
        method = c("Broyden", "Newton"),
        global = c("dbldog", "pwldog", "qline", "gline", "none"),
        xscalm = c("fixed", "auto"),
        control = list()
)
```

Arguments

<code>x</code>	A list or numeric vector of starting estimates.
<code>fn</code>	A function of <code>x</code> returning the function values.
<code>jac</code>	A function to return the Jacobian for the <code>fn</code> function. If not supplied numerical derivatives will be used.
<code>...</code>	Further arguments to be passed to <code>fn</code> and <code>jac</code> .
<code>method</code>	The method used for finding a solution. See ‘Details’.
<code>global</code>	The global strategy to apply. See ‘Details’.
<code>xscalm</code>	The type of <code>x</code> scaling to use. See ‘Details’.
<code>control</code>	A list of control parameters. See ‘Details’.

Details

The algorithms implemented in `nleqslv` are based on Dennis and Schnabel (1996).

Method Broyden starts with a computed Jacobian of the function and then updates this Jacobian after each successful iteration using the so-called Broyden update. This method often shows super linear convergence towards a solution. When `nleqslv` determines that it cannot continue with the current Broyden matrix it will compute a new Jacobian.

Method Newton calculates a Jacobian of the function `fn` at each iteration. Close to a solution this method will usually show quadratic convergence.

Both methods apply a so-called (backtracking) global strategy to find a better (more acceptable) iterate. The function criterion used by the algorithm is the sum of squares of the function values and “acceptable” means sufficient decrease of the current function criterion value compared to that of the previous iteration. A comprehensive discussion of these issues can be found in Dennis and Schnabel (1996). Both methods apply a QR-decomposition to the Jacobian as implemented in Lapack. The Broyden method applies a rank-1 update to the Jacobian at the end of each iteration and uses an algorithm described for example in Golub and Van Loan (1996) and in Reichel and Gragg (1990). The code used in the package is taken from the opensource package `QRupdate`.

The `global` argument indicates which global strategy to use or to use no global strategy

`qline` a quadratic linesearch

`gline` a geometric linesearch

`dbldog` a trust region method using the double dogleg method as described in Dennis and Schnabel (1996)

`pwldog` a trust region method using the Powell dogleg method as developed by Powell (1970).

`none` Only a pure local Newton or Broyden iteration is used. The maximum stepsize (see below) is taken into account. The default maximum number of iterations (see below) is set to 20.

The double dogleg method is the default global strategy employed by this package.

The parameters `x` may be scaled during the search for a zero of `fn`. The `xscalm` argument provides two possibilities for scaling

`fixed` the scaling factors are set to the values supplied in the `control` argument and remain unchanged during the iterations. The scaling factor of any element of `x` should be set to the inverse of the typical value of that element of `x`, ensuring that all elements of `x` are approximately equal in size.

`auto` the scaling factors are calculated from the euclidean norms of the columns of the jacobian matrix. When a new Jacobian is computed, the scaling values will be set to the euclidean norm of the corresponding column if that is larger than the current scaling value. Thus the scaling values will not decrease during the iteration. This is the method described in Moré (1978). Usually manual scaling is preferable.

The algorithm cannot cope with a singular or ill-conditioned Jacobian, in contrast to previous versions of this package. When a Jacobian becomes singular or very ill-conditioned the algorithm will return an error. A Jacobian is considered to be very ill-conditioned when the estimated inverse condition is less than or equal to the machine precision.

When the function to be solved returns non-finite values for a parameter vector x and the algorithm is *not* evaluating a numerical jacobian, then the non-finite value will be replaced by a large number forcing the algorithm to backtrack, i.e. decrease the linesearch factor or decrease the trust region radius. When the algorithm is evaluating a numerical jacobian, it will stop with an error message on detecting non-finite function values.

The control argument is a list that can supply any of the following components:

`xtol` The relative steplength tolerance. When the relative steplength of all scaled x values is smaller than this value convergence is declared. The default value is 10^{-8} .

`ftol` The function value tolerance. Convergence is declared when the largest absolute function value is smaller than `ftol`. The default value is 10^{-8} .

`btol` The backtracking tolerance. When the relative steplength in a backtracking step to find an acceptable point is smaller than the backtracking tolerance, the backtracking is terminated. In the Broyden method a new Jacobian will be calculated if the Jacobian is outdated. The default value is 10^{-3} .

`sigma` Reduction factor for the geometric linesearch. The default value is 0.5.

`scalex` a vector of scaling values for the parameters. The inverse of a scale value is an indication of the size of a parameter. The default value is 1.0 for all scale values.

`maxit` The maximum number of major iterations. The default value is 150 if a global strategy has been specified. If no global strategy has been specified the default is 20.

`trace` Non-negative integer. A value of 1 will give a detailed report of the progress of the iteration. For a description see [Iteration report](#).

`chkjac` A logical value indicating whether to check an analytical jacobian, if supplied. The default value is FALSE. The first 10 errors are printed. The code for this check is derived from the code in Bouaricha and Schnabel (1997).

`delta` Initial (scaled) trust region radius. A value of -1.0 is replaced by the length of the Cauchy step in the initial point. A value of -2.0 is replaced by the length of the Newton step in the initial point. Any value less than or equal to 0 and not equal to -2.0 , will be replaced by -1.0 ; the algorithm will thus start with the length of the Cauchy step in the initial point. If it is positive it will be set to the smaller of the value supplied or the maximum stepsize. The default is -2.0 .

`stepmax` Maximum scaled stepsize. If this is negative then the maximum stepsize is set to the largest positive representable number. The default is -1.0 , so there is no default maximum stepsize.

Value

A list containing components

x	final values for x
fvec	function values
termcd	termination code as integer. The values returned are <ol style="list-style-type: none"> 1 Function criterion is near zero. Convergence of function values has been achieved. 2 x-values within tolerance. This means that the relative distance between two consecutive x-values is smaller than <code>xtol</code>. 3 No better point found. This means that the algorithm has stalled and cannot find an acceptable new point. This may or may not indicate acceptably small function values. 4 Iteration limit <code>maxit</code> exceeded. 5 Jacobian is too ill-conditioned. 6 Jacobian is singular. -10 Analytical Jacobian is most likely incorrect.
message	a string describing the termination code
scalex	a vector containing the scaling factors, which will be the final values when automatic scaling was selected
njcmt	number of jacobian evaluations
nfcmt	number of function evaluations, excluding those required for calculating a jacobian

Warning

You cannot use this function recursively. Thus function `fn` should not in its turn call `nleqslv`.

References

- Bouaricha, A. and Schnabel, R.B. (1997), Algorithm 768: TENSOLVE: A Software Package for Solving Systems of Nonlinear Equations and Nonlinear Least-squares Problems Using Tensor Methods, *Transactions on Mathematical Software*, **23**, 2, pp. 174–195.
- Dennis, J.E. Jr and Schnabel, R.B. (1996), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Siam.
- Moré, J.J. (1978), The Levenberg-Marquardt Algorithm, Implementation and Theory, In *Numerical Analysis*, G.A. Watson (Ed.), Lecture Notes in Mathematics 630, Springer-Verlag, pp. 105–116.
- Golub, G.H and C.F. Van Loan (1996), *Matrix Computations* (3rd edition), The John Hopkins University Press.
- Powell, M.J.D. (1970), A hybrid method for nonlinear algebraic equations, In *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz (Ed.), Gordon & Breach.
- Powell, M.J.D. (1970), A Fortran subroutine for solving systems nonlinear equations, In *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz (Ed.), Gordon & Breach.

QRupdate: a Fortran library for fast updating of QR and Cholesky decompositions, <http://sourceforge.net/projects/qrupdate/>.

Reichel, L. and W.B. Gragg (1990), Algorithm 686: FORTRAN subroutines for updating the QR decomposition, *ACM Trans. Math. Softw.*, **16**, 4, pp. 369–377.

Examples

```
# Dennis Schnabel example 6.5.1 page 149
dslnex <- function(x) {
  y <- numeric(2)
  y[1] <- x[1]^2 + x[2]^2 - 2
  y[2] <- exp(x[1]-1) + x[2]^3 - 2
  y
}

jacdsln <- function(x) {
  n <- length(x)
  Df <- matrix(numeric(n*n),n,n)
  Df[1,1] <- 2*x[1]
  Df[1,2] <- 2*x[2]
  Df[2,1] <- exp(x[1]-1)
  Df[2,2] <- 3*x[2]^2

  Df
}

BADjacdsln <- function(x) {
  n <- length(x)
  Df <- matrix(numeric(n*n),n,n)
  Df[1,1] <- 4*x[1]
  Df[1,2] <- 2*x[2]
  Df[2,1] <- exp(x[1]-1)
  Df[2,2] <- 5*x[2]^2

  Df
}

xstart <- c(2,0.5)
fstart <- dslnex(xstart)
xstart
fstart

# a solution is c(1,1)

nleqslv(xstart, dslnex, control=list(btol=.01))

# Cauchy start
nleqslv(xstart, dslnex, control=list(trace=1,btol=.01,delta=-1.0))

# Newton start
nleqslv(xstart, dslnex, control=list(trace=1,btol=.01,delta=-2.0))
```

```

## Not run:
# no global strategy but limit stepsize
# but look carefully: a different solution is found
nleqslv(xstart, dslnex, method="Newton", global="none", control=list(trace=1,stepmax=5))

# but if the stepsize is limited even more the c(1,1) solution is found
nleqslv(xstart, dslnex, method="Newton", global="none", control=list(trace=1,stepmax=2))

# Broyden also finds the c(1,1) solution when the stepsize is limited
nleqslv(xstart, dslnex, jacdsln, method="Broyden", global="none", control=list(trace=1,stepmax=2))

## End(Not run)

```

nleqslv-iterationreport

Detailed iteration report of nleqslv

Description

The format of the iteration report provided by nleqslv when the trace component of the control parameter has been set to 1.

Details

All iteration reports consist of a series of columns with a header summarising the contents. Common column headers are

Iter Iteration counter

Jac Jacobian type. The jacobian type is indicated by N for a Newton jacobian or B for a Broyden updated matrix; optionally followed by the letter s indicating a totally singular matrix or the letter i indicating an ill-conditioned matrix. Unless the jacobian is singular, the jacobian type is followed by an estimate of the inverse condition number of the jacobian in parentheses as computed by Lapack. This column will be blank when backtracking is active.

Fnorm euclidean norm of function values / 2

Largest |f| infinity norm of $f(x)$ at the current point

A sample iteration report for qline is (some intercolumn space has been removed to make the table fit)

Iter	Jac	Lambda	Ftarg	Fnorm	Largest f
0				2.886812e+00	2.250000e+00
1	N(9.6e-03)	1.0000	2.886235e+00	5.787362e+05	1.070841e+03
1		0.1000	2.886754e+00	9.857947e+00	3.214799e+00
1		0.0100	2.886806e+00	2.866321e+00	2.237878e+00
2	B(2.2e-02)	1.0000	2.865748e+00	4.541965e+03	9.341610e+01

The column headed by Lambda shows the value of the linesearch parameter. The column headed by Ftarg follows from a sufficient decrease requirement and is the value below which Fnorm must drop if the current step is to be accepted.

The iteration report for none is almost the same as the above report, except that the column labelled Ftarg is omitted. The column Lambda gives the ratio of the maximum stepsize and the length of the step taken by the algorithm. It is either 1.0 exactly or smaller.

A sample iteration report for dbldog is (some intercolumn space has been removed to make the table fit)

Iter	Jac	Lambda	Eta	Dlt0	Dltn	Fnorm	Largest f
0						2.886812e+00	2.250000e+00
1	N(9.6e-03)	C	0.9544	0.4671	0.9343	1.699715e-01	5.421673e-01
1		W 0.0833	0.9544	0.9343	0.4671	1.699715e-01	5.421673e-01
2	B(1.1e-02)	W 0.1154	0.4851	0.4671	0.4671	1.277667e-01	5.043571e-01
3	B(7.3e-02)	W 0.7879	0.7289	0.4671	0.0759	5.067893e-01	7.973542e-01
3		C	0.7289	0.0759	0.1519	5.440250e-02	2.726084e-01
4	B(8.3e-02)	W 0.5307	0.3271	0.1519	0.3037	3.576547e-02	2.657553e-01
5	B(1.8e-01)	N	0.6674	0.2191	0.4383	6.566182e-03	8.555110e-02
6	B(1.8e-01)	N	0.9801	0.0376	0.0752	4.921645e-04	3.094104e-02
7	B(1.9e-01)	N	0.7981	0.0157	0.0313	4.960629e-06	2.826064e-03
8	B(1.6e-01)	N	0.3942	0.0029	0.0058	1.545503e-08	1.757498e-04
9	B(1.5e-01)	N	0.6536	0.0001	0.0003	2.968676e-11	5.983765e-06
10	B(1.5e-01)	N	0.4730	0.0000	0.0000	4.741792e-14	2.198380e-07
11	B(1.5e-01)	N	0.9787	0.0000	0.0000	6.451792e-19	8.118586e-10

After the column for the jacobian the letters indicate the following

C a fraction (≤ 1.0) of the Cauchy or steepest descent step is taken where the fraction is the ratio of the trust region radius and the Cauchy steplength.

W a convex combination of the Cauchy and eta*(Newton step) is taken. The number in the column headed by Lambda is the weight of the partial Newton step.

P a fraction (≥ 1.0) of the partial Newton step, equal to eta*(Newton step), is taken where the fraction is the ratio of the trust region radius and the partial Newton steplength.

N a normal full Newton step is taken.

The number in the column headed by Eta is calculated from an upper limit on the ratio of the length of the steepest descent direction and the length of the Newton step. See Dennis and Schnabel (1996) pp.139ff for the details.

The column headed by Dlt0 gives the size of the trust region at the start of the current iteration. The column headed by Dltn gives the size of the trust region when the current step has been accepted by the dogleg algorithm. The size of the trust region is decreased when the actual reduction of the function value norm does not agree well with the predicted reduction from the linear approximation of the function. And increased when the actual and predicted reduction are sufficiently close.

Normally the initial trust region radius is the same as the final trust region radius of the previous iteration but the size of the trust region is restricted by the size of the current Newton step. So when full Newton steps are being taken, the trust region radius at the start of an iteration may be less than the final value of the previous iteration.

A sample iteration report for pwldog is (some intercolumn space has been removed to make the table fit)

Iter	Jac	Lambda	Dlt0	Dltn	Fnorm	Largest	f
0					2.886812e+00	2.250000e+00	
1	N(9.6e-03)	C	0.4671	0.9343	1.699715e-01	5.421673e-01	
1		W 0.0794	0.9343	0.4671	1.699715e-01	5.421673e-01	
2	B(1.1e-02)	W 0.0559	0.4671	0.4671	1.205661e-01	4.890487e-01	
3	B(7.3e-02)	W 0.5662	0.4671	0.0960	4.119560e-01	7.254441e-01	
3		W 0.0237	0.0960	0.1921	4.426507e-02	2.139252e-01	
4	B(8.8e-02)	W 0.2306	0.1921	0.3842	2.303135e-02	2.143943e-01	
4		W 0.4769	0.3842	0.1921	2.303135e-02	2.143943e-01	
5	B(1.9e-01)	N	0.1375	0.2750	8.014508e-04	3.681498e-02	

This is much simpler than the double dogleg report, since the (single) dogleg takes either a steepest descent step, a convex combination of the steepest descent and Newton directions or a full Newton step. The number in the column Lambda is the weight of the Newton step. It is a special case of the double dogleg method with eta equal to 1.

See Also

[nleqslv](#)

Index

*Topic **nonlinear**

nleqslv, [2](#)

*Topic **optimize**

nleqslv, [2](#)

*Topic **package**

nleqslv-package, [2](#)

Iteration report, [4](#)

Iteration report

(nleqslv-iterationreport), [7](#)

nleqslv-package, [2](#)

nleqslv.Intro (nleqslv-package), [2](#)

nleqslv, [2](#), [9](#)

nleqslv-iterationreport, [7](#)