L-BFGS-B

3.0

Generated by Doxygen 1.8.13

Contents

1 L-BFGS-B

Software for Large-scale Bound-constrained Optimization

L-BFGS-B is a limited-memory quasi-Newton code for bound-constrained optimization, i.e., for problems where the only constraints are of the form `l <= x <= u`. It is intended for problems in which information on the Hessian matrix is difficult to obtain, or for large dense problems. L-BFGS-B can also be used for unconstrained problems, and in this case performs similarly to its predecessor, algorithm L-BFGS (Harwell routine VA15). The algorithm is implemented in Fortran 77.

Authors

- Ciyou Zhu
- Richard Byrd
- Jorge Nocedal
- Jose Luis Morales

Related Publications

- R. H. Byrd, P. Lu, J. Nocedal and C. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization (1995), SIAM Journal on Scientific and Statistical Computing, Vol. 16, Num. 5, pp. 1190-1208
- C. Zhu, R. H. Byrd and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization (1997), ACM Transactions on Mathematical Software, Vol. 23, Num. 4, pp. 550-560
- J.L. Morales and J. Nocedal. L-BFGS-B: Remark on Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization (2011), ACM Transactions on Mathematical Software, Vol. 38, Num. 1
- R. H. Byrd, J. Nocedal and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods (1994), Mathematical Programming, Vol. 63, pp. 129-156

Note that the subspace minimization in the LBFGSpp implementation is an exact minimization subject to the bounds, based on the BOXCQP algorithm:

- C. Voglis and I. E. Lagaris, BOXCQP: An Algorithm for Bound Constrained Convex Quadratic Problems (2004), 1st International Conference "From Scientific Computing to Computational Engineering", Athens, Greece

For an eagle-eye overview of L-BFGS-B and the genealogy BFGS->L-BFGS->L-BFGS-B, see Henao's Master's thesis.

Related Software

- wilmerhenao/L-BFGS-B-NS: An L-BFGS-B-NS Optimizer for Non-Smooth Functions
- pcarbo/lbfgsb-matlab: A MATLAB interface for L-BFGS-B
- bgranzow/L-BFGS-B: A pure Matlab implementation of L-BFGS-B (LBFGSB)
- constantino-garcia/lbfgsb_cpp_wrapper: A simple C++ wrapper around the original Fortran L-BGSG-B routine
- yixuan/LBFGSpp: A header-only C++ library for L-BFGS and L-BFGS-B algorithms
- chokkan/liblbfgs: libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)
- mkobos/lbfgsb_wrapper: Java wrapper for the Fortran L-BFGS-B algorithm
- yuhonglin/Lbfgsb.jl: A Julia wrapper of the l-bfgs-b fortran library
- Gnimuc/LBFGSB.jl: Julia wrapper for L-BFGS-B Nonlinear Optimization Code
- afbarnard/go-lbfgsb: L-BFGS-B optimization for Go, C, Fortran 2003
- nepluno/lbfgsb-gpu: An open source library for the GPU-implementation of L-BFGS-B algorithm
- Chris00/L-BFGS-ocaml: OCaml bindings for L-BFGS
- dwicke/L-BFGS-B-Lua: L-BFGS-B lua wrapper around a L-BFGS-B C implementation
- avieira/python_lbfgsb: Pure Python-based L-BFGS-B implementation
- ybyygu/rust-lbfgsb: Ergonomic bindings to L-BFGS-B code for Rust
- lbfgsb3c: Limited Memory BFGS Minimizer with Bounds on Parameters with optim() 'C' Interface for R

Notes on this repository

I (J. Schilling) took the freedom to

- put the L-BFGS-B code obtained from the original website up in this repository,
- divide the subroutines and functions into separate files,
- convert parts of the documentation into a format understandable to doxygen and
- adjust the Makefile to accomodate the separate files and additionally generate a statically linked liblbfgsb.a library.

The current release is version 3.0. The distribution file was last changed on 02/08/11.

This work was in no way intending to infringe any copyrights or take credit for others' work. Feel free to contact me at any time in case you noticed something against the rules. Above documentation is obtained from the archived version of the original manual.

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

3 File Documentation

3.1 active.f File Reference

Functions/Subroutines

- subroutine active (n, l, u, nbd, x, iwhere, iprint, prjctd, cnstnd, boxed)

  This subroutine initializes iwhere and projects the initial x to the feasible set if necessary.

3.1.1 Function/Subroutine Documentation

3.1.1.1 active()

```
subroutine active (
             integer n,
             double precision, dimension(n) l,
             double precision, dimension(n) u,
             integer, dimension(n) nbd,
             double precision, dimension(n) x,
             integer, dimension(n) iwhere,
             integer iprint,
             logical prjctd,
             logical cnstnd,
             logical boxed )
```

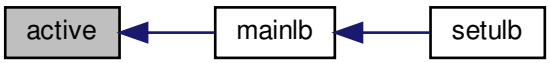
This subroutine initializes iwhere and projects the initial x to the feasible set if necessary.

Parameters

n	number of parameters
l	lower bounds on parameters
u	upper bounds on parameters
nbd	indicates which bounds are present
x	position
iwhere	On entry iwhere is unspecified. On exit: iwhere(i)= • -1 if x(i) has no bounds • 3 if l(i)=u(i), • 0 otherwise. In cauchy, iwhere is given finer gradations.
iprint	console output flag
prjctd	TODO
cnstnd	TODO
boxed	TODO

Definition at line 34 of file active.f.

Here is the caller graph for this function:



3.2 bmv.f File Reference

Functions/Subroutines

- subroutine bmv (m, sy, wt, col, v, p, info)

  This subroutine computes the product of the 2m x 2m middle matrix in the compact L-BFGS formula of B and a 2m vector v.

3.2.1 Function/Subroutine Documentation

3.2.1.1 bmv()

```
subroutine bmv (
             integer m,
             double precision, dimension(m, m) sy,
             double precision, dimension(m, m) wt,
             integer col,
             double precision, dimension(2*col) v,
             double precision, dimension(2*col) p,
             integer info )
```

This subroutine computes the product of the 2m x 2m middle matrix in the compact L-BFGS formula of B and a 2m vector v; it returns the product in p.

Parameters

m	On entry m is the maximum number of variable metric corrections used to define the limited memory matrix. On exit m is unchanged.
sy	On entry sy specifies the matrix S'Y. On exit sy is unchanged.
wt	On entry wt specifies the upper triangular matrix J' which is the Cholesky factor of (thetaS'S+LD^(-1)L'). On exit wt is unchanged.
col	On entry col specifies the number of s-vectors (or y-vectors) stored in the compact L-BFGS formula. On exit col is unchanged.
v	On entry v specifies vector v. On exit v is unchanged.

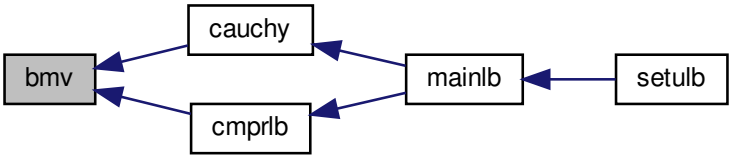
Parameters

p	On entry p is unspecified. On exit p is the product Mv.
info	On entry info is unspecified. On exit info = • 0 for normal return, • nonzero for abnormal return when the system to be solved by dtrsl is singular.

Definition at line 36 of file bmv.f.

Here is the call graph for this function:

Here is the caller graph for this function:



3.3 cauchy.f File Reference

Functions/Subroutines

- subroutine cauchy (n, x, l, u, nbd, g, iorder, iwhere, t, d, xcp, m, wy, ws, sy, wt, theta, col, head, p, c, wbp, v, nseg, iprint, sbgnrm, info, epsmch)

  Compute the Generalized Cauchy Point along the projected gradient direction.

3.3.1 Function/Subroutine Documentation

3.3.1.1 cauchy()

```
subroutine cauchy (
             integer n,
             double precision, dimension(n) x,
             double precision, dimension(n) l,
             double precision, dimension(n) u,
             integer, dimension(n) nbd,
             double precision, dimension(n) g,
             integer, dimension(n) iorder,
             integer, dimension(n) iwhere,
             double precision, dimension(n) t,
             double precision, dimension(n) d,
             double precision, dimension(n) xcp,
             integer m,
             double precision, dimension(n, col) wy,
             double precision, dimension(n, col) ws,
             double precision, dimension(m, m) sy,
             double precision, dimension(m, m) wt,
             double precision theta,
             integer col,
             integer head,
             double precision, dimension(2*m) p,
             double precision, dimension(2*m) c,
             double precision, dimension(2*m) wbp,
             double precision, dimension(2*m) v,
             integer nseg,
             integer iprint,
             double precision sbgnrm,
             integer info,
             double precision epsmch )
```

For given x, l, u, g (with sbgnrm $>$ 0), and a limited memory BFGS matrix B defined in terms of matrices WY, WS, WT, and scalars head, col, and theta, this subroutine computes the generalized Cauchy point (GCP), defined as the first local minimizer of the quadratic

```
Q(x + s) = g's + 1/2 s'Bs
```

along the projected gradient direction P(x-tg,l,u). The routine returns the GCP in xcp.

Parameters

n	On entry n is the dimension of the problem. On exit n is unchanged.
x	On entry x is the starting point for the GCP computation. On exit x is unchanged.
l	On entry l is the lower bound of x. On exit l is unchanged.
u	On entry u is the upper bound of x. On exit u is unchanged.
nbd	On entry nbd represents the type of bounds imposed on the variables, and must be specified as follows: nbd(i)= • 0 if x(i) is unbounded, • 1 if x(i) has only a lower bound, • 2 if x(i) has both lower and upper bounds, and • 3 if x(i) has only an upper bound.
	On exit nbd is unchanged.

Parameters

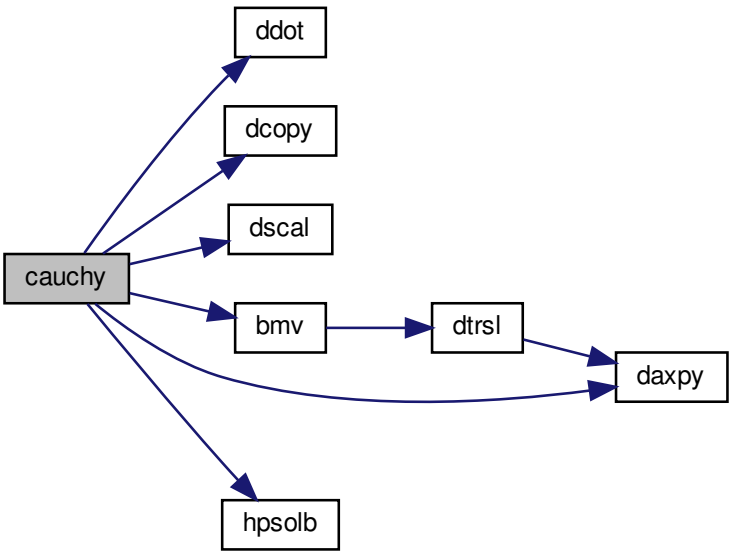
g	On entry g is the gradient of f(x). g must be a nonzero vector. On exit g is unchanged.
iorder	iorder will be used to store the breakpoints in the piecewise linear path and free variables encountered. On exit, • iorder(1),...,iorder(nleft) are indices of breakpoints which have not been encountered; • iorder(nleft+1),...,iorder(nbreak) are indices of encountered breakpoints; and • iorder(nfree),...,iorder(n) are indices of variables which have no bound constraits along the search direction.
iwhere	On entry iwhere indicates only the permanently fixed (iwhere=3) or free (iwhere= -1) components of x. On exit iwhere records the status of the current x variables. iwhere(i)= • -3 if x(i) is free and has bounds, but is not moved • 0 if x(i) is free and has bounds, and is moved • 1 if x(i) is fixed at l(i), and l(i) .ne. u(i) • 2 if x(i) is fixed at u(i), and u(i) .ne. l(i) • 3 if x(i) is always fixed, i.e., u(i)=x(i)=l(i) • -1 if x(i) is always free, i.e., it has no bounds.
t	working array; will be used to store the break points.
d	the Cauchy direction P(x-tg)-x
xcp	returns the GCP on exit
m	On entry m is the maximum number of variable metric corrections used to define the limited memory matrix. On exit m is unchanged.
ws	On entry this stores S, a set of s-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wy	On entry this stores Y, a set of y-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
sy	On entry this stores S'Y, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wt	On entry this stores the Cholesky factorization of (theta∗S'S+LD^(-1)L'), that defines the limited memory BFGS matrix. On exit this array is unchanged.
theta	On entry theta is the scaling factor specifying B_0 = theta I. On exit theta is unchanged.
col	On entry col is the actual number of variable metric corrections stored so far. On exit col is unchanged.
head	On entry head is the location of the first s-vector (or y-vector) in S (or Y). On exit col is unchanged.
p	will be used to store the vector p = W^(T)d.
c	will be used to store the vector c = W^(T)(xcp-x).
wbp	will be used to store the row of W corresponding to a breakpoint.
v	working array
nseg	On exit nseg records the number of quadratic segments explored in searching for the GCP.

Parameters

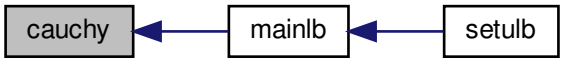
iprint	variable that must be set by the user. It controls the frequency and type of output generated: • iprint<0 no output is generated; • iprint=0 print only one line at the last iteration; • 0<iprint<99 print also f and |proj g| every iprint iterations; • iprint=99 print details of every iteration except n-vectors; • iprint=100 print also the changes of active set and final x; • iprint>100 print details of every iteration including x and g; When iprint > 0, the file iterate.dat will be created to summarize the iteration.
sbgnrm	On entry sbgnrm is the norm of the projected gradient at x. On exit sbgnrm is unchanged.
info	On entry info is 0. On exit info = • 0 for normal return, • = nonzero for abnormal return when the the system used in routine bmv is singular.
epsmch	machine precision epsilon

Definition at line 133 of file cauchy.f.

Here is the call graph for this function:



Here is the caller graph for this function:



3.4 cmprlb.f File Reference

Functions/Subroutines

- subroutine cmprlb (n, m, x, g, ws, wy, sy, wt, z, r, wa, index, theta, col, head, nfree, cnstnd, info)

  This subroutine computes r=-Z'B(xcp-xk)-Z'g by using wa(2m+1)=W'(xcp-x) from subroutine cauchy.

3.4.1 Function/Subroutine Documentation

3.4.1.1 cmprlb()

```
subroutine cmprlb (
             integer n,
             integer m,
             double precision, dimension(n) x,
             double precision, dimension(n) g,
             double precision, dimension(n, m) ws,
             double precision, dimension(n, m) wy,
             double precision, dimension(m, m) sy,
             double precision, dimension(m, m) wt,
             double precision, dimension(n) z,
             double precision, dimension(n) r,
             double precision, dimension(4*m) wa,
             integer, dimension(n) index,
             double precision theta,
             integer col,
             integer head,
             integer nfree,
             logical cnstnd,
             integer info )
```

This subroutine computes r=-Z'B(xcp-xk)-Z'g by using wa(2m+1)=W'(xcp-x) from subroutine cauchy.

Parameters

n	number of parameters
m	history size of Hessian approximation
x	position
g	gradient

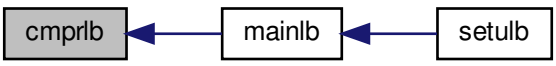
Parameters

ws	part of L-BFGS matrix
wy	part of L-BFGS matrix
sy	part of L-BFGS matrix
wt	part of L-BFGS matrix
z	TODO
r	TODO
wa	TODO
index	TODO
theta	TODO
col	TODO
head	TODO
nfree	TODO
cnstnd	TODO
info	TODO

Definition at line 29 of file cmprlb.f.

Here is the call graph for this function:

Here is the caller graph for this function:



3.5 daxpy.f File Reference

Functions/Subroutines

- subroutine daxpy (n, da, dx, incx, dy, incy)

  constant times a vector plus a vector.

3.5.1 Function/Subroutine Documentation

3.5.1.1 daxpy()

```
subroutine daxpy (
             integer n,
             double precision da,
             double precision, dimension(*) dx,
             integer incx,
             double precision, dimension(*) dy,
             integer incy )
```

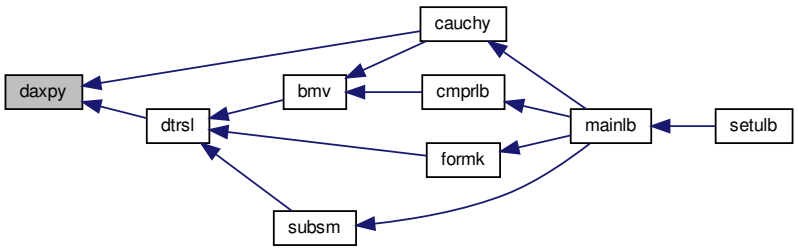
constant times a vector plus a vector. uses unrolled loops for increments equal to one.

Parameters

n	dimensionality of vectors
da	constant for scaling dx
dx	vector to be scaled and added to dy
incx	spacing between elements in dx
dy	target vector to which da∗dx gets added element-wise
incy	spacing between elements in dy

Definition at line 15 of file daxpy.f.

Here is the caller graph for this function:



3.6 dcopy.f File Reference

Functions/Subroutines

- subroutine dcopy (n, dx, incx, dy, incy)

  copies a vector, x, to a vector, y.

3.6.1 Function/Subroutine Documentation

3.6.1.1 dcopy()

```
subroutine dcopy (
             integer n,
             double precision, dimension(*) dx,
             integer incx,
             double precision, dimension(*) dy,
             integer incy )
```

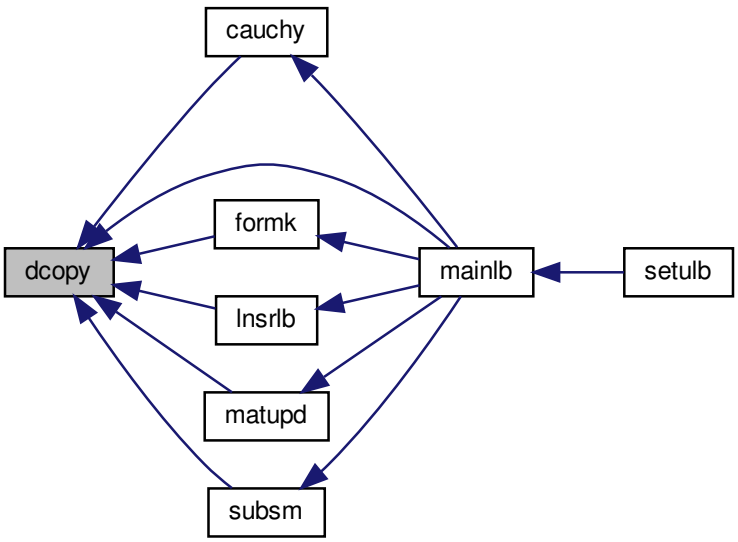
copies a vector, x, to a vector, y. uses unrolled loops for increments equal to one.

Parameters

n	dimensionality of vectors
dx	source vector
incx	spacing of elements in dx
dy	target vector
incy	spacing of elements in dy

Definition at line 14 of file dcopy.f.

Here is the caller graph for this function:



3.7 dcsrch.f File Reference

Functions/Subroutines

- subroutine dcsrch (f, g, stp, ftol, gtol, xtol, stpmin, stpmax, task, isave, dsave)

  This subroutine finds a step that satisfies a sufficient decrease condition and a curvature condition.

3.7.1 Function/Subroutine Documentation

3.7.1.1 dcsrch()

```
subroutine dcsrch (
             double precision f,
             double precision g,
             double precision stp,
             double precision ftol,
             double precision gtol,
             double precision xtol,
             double precision stpmin,
             double precision stpmax,
             character*(*) task,
             integer, dimension(2) isave,
             double precision, dimension(13) dsave )
```

This subroutine finds a step that satisfies a sufficient decrease condition and a curvature condition.

Each call of the subroutine updates an interval with endpoints stx and sty. The interval is initially chosen so that it contains a minimizer of the modified function

```
psi(stp) = f(stp) - f(0) - ftol*stp*f'(0).
```

If psi(stp) <= 0 and f'(stp) >= 0 for some step, then the interval is chosen so that it contains a minimizer of f.

The algorithm is designed to find a step that satisfies the sufficient decrease condition

```
f(stp) <= f(0) + ftol*stp*f'(0),
```

and the curvature condition

```
abs(f'(stp)) <= gtol*abs(f'(0)).
```

If ftol is less than gtol and if, for example, the function is bounded below, then there is always a step which satisfies both conditions.

If no step can be found that satisfies both conditions, then the algorithm stops with a warning. In this case stp only satisfies the sufficient decrease condition.

A typical invocation of dcsrch has the following outline:

```
   task = 'START'
10 continue
   call dcsrch( ... )
   if (task .eq. 'FG') then
     evaluate the function and the gradient at stp
   goto 10
   end if
```

NOTE: The user must no alter work arrays between calls.

Parameters

f	On initial entry f is the value of the function at 0. On subsequent entries f is the value of the function at stp. On exit f is the value of the function at stp.
g	On initial entry g is the derivative of the function at 0. On subsequent entries g is the derivative of the function at stp. On exit g is the derivative of the function at stp.
stp	On entry stp is the current estimate of a satisfactory step. On initial entry, a positive initial estimate must be provided. On exit stp is the current estimate of a satisfactory step if task = 'FG'. If task = 'CONV' then stp satisfies the sufficient decrease and curvature condition.
ftol	On entry ftol specifies a nonnegative tolerance for the sufficient decrease condition. On exit ftol is unchanged.
gtol	On entry gtol specifies a nonnegative tolerance for the curvature condition. On exit gtol is unchanged.
xtol	On entry xtol specifies a nonnegative relative tolerance for an acceptable step. The subroutine exits with a warning if the relative difference between sty and stx is less than xtol. On exit xtol is unchanged.
stpmin	On entry stpmin is a nonnegative lower bound for the step. On exit stpmin is unchanged.
stpmax	On entry stpmax is a nonnegative upper bound for the step. On exit stpmax is unchanged.
task	On initial entry task must be set to 'START'. On exit task indicates the required action: • If task(1:2) = 'FG' then evaluate the function and derivative at stp and call dcsrch again. • If task(1:4) = 'CONV' then the search is successful. • If task(1:4) = 'WARN' then the subroutine is not able to satisfy the convergence conditions. The exit value of stp contains the best point found during the search. • If task(1:5) = 'ERROR' then there is an error in the input arguments. On exit with convergence, a warning or an error, the variable task contains additional information.
isave	work array
dsave	work array

Definition at line 96 of file dcsrch.f.

Here is the call graph for this function:

Here is the caller graph for this function:

3.8 dcstep.f File Reference

Functions/Subroutines

- subroutine dcstep (stx, fx, dx, sty, fy, dy, stp, fp, dp, brackt, stpmin, stpmax)

  This subroutine computes a safeguarded step for a search procedure and updates an interval that contains a step that satisfies a sufficient decrease and a curvature condition.

3.8.1 Function/Subroutine Documentation

3.8.1.1 dcstep()

```
subroutine dcstep (
             double precision stx,
             double precision fx,
             double precision dx,
             double precision sty,
             double precision fy,
             double precision dy,
             double precision stp,
             double precision fp,
             double precision dp,
             logical brackt,
             double precision stpmin,
             double precision stpmax )
```

This subroutine computes a safeguarded step for a search procedure and updates an interval that contains a step that satisfies a sufficient decrease and a curvature condition.

The parameter stx contains the step with the least function value. If brackt is set to .true. then a minimizer has been bracketed in an interval with endpoints stx and sty. The parameter stp contains the current step. The subroutine assumes that if brackt is set to .true. then

```
min(stx,sty) < stp < max(stx,sty),
```

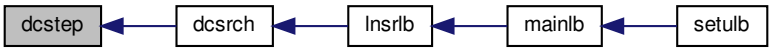
and that the derivative at stx is negative in the direction of the step.

Parameters

stx	On entry stx is the best step obtained so far and is an endpoint of the interval that contains the minimizer. On exit stx is the updated best step.
fx	On entry fx is the function at stx. On exit fx is the function at stx.
dx	On entry dx is the derivative of the function at stx. The derivative must be negative in the direction of the step, that is, dx and stp - stx must have opposite signs. On exit dx is the derivative of the function at stx.
sty	On entry sty is the second endpoint of the interval that contains the minimizer. On exit sty is the updated endpoint of the interval that contains the minimizer.
fy	On entry fy is the function at sty. On exit fy is the function at sty.
dy	On entry dy is the derivative of the function at sty. On exit dy is the derivative of the function at the exit sty.
stp	On entry stp is the current step. If brackt is set to .true. then on input stp must be between stx and sty. On exit stp is a new trial step.
fp	On entry fp is the function at stp. On exit fp is unchanged.
dp	On entry dp is the the derivative of the function at stp. On exit dp is unchanged.
brackt	On entry brackt specifies if a minimizer has been bracketed. Initially brackt must be set to .false. On exit brackt specifies if a minimizer has been bracketed. When a minimizer is bracketed brackt is set to .true.
stpmin	On entry stpmin is a lower bound for the step. On exit stpmin is unchanged.
stpmax	On entry stpmax is an upper bound for the step. On exit stpmax is unchanged.

Definition at line 68 of file dcstep.f.

Here is the caller graph for this function:



3.9 ddot.f File Reference

Functions/Subroutines

- double precision function ddot (n, dx, incx, dy, incy)

  forms the dot product of two vectors.

3.9.1 Function/Subroutine Documentation

3.9.1.1 ddot()

```
double precision function ddot (
             integer n,
             double precision, dimension(*) dx,
             integer incx,
             double precision, dimension(*) dy,
             integer incy )
```

forms the dot product of two vectors. uses unrolled loops for increments equal to one.

Parameters

n	dimensionality of vectors
dx	first vector
incx	spacing of elements in dx
dy	second vector
incy	spacing of elements in dy

Returns

dot product of dx and dy

Definition at line 15 of file ddot.f.

Here is the caller graph for this function:

3.10 dpofa.f File Reference

Functions/Subroutines

- subroutine dpofa (a, lda, n, info)

  factors a double precision symmetric positive definite matrix.

3.10.1 Function/Subroutine Documentation

3.10.1.1 dpofa()

```
subroutine dpofa (
             double precision, dimension(lda,*) a,
             integer lda,
             integer n,
             integer info )
```

factors a double precision symmetric positive definite matrix.

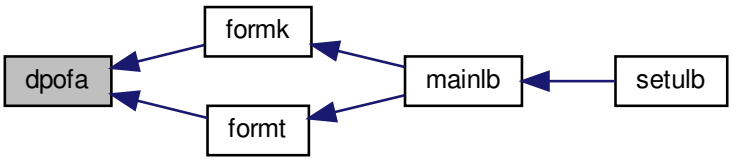
dpofa is usually called by dpoco, but it can be called directly with a saving in time if rcond is not needed. (time for dpoco) = (1 + 18/n)∗(time for dpofa) .

Parameters

a	On entry, this is the symmetric matrix to be factored. Only the diagonal and upper triangle are used. On exit this is an upper triangular matrix r so that a = trans(r)∗r where trans(r) is the transpose. The strict lower triangle is unaltered. If info .ne. 0 , the factorization is not complete.
lda	On entry, this is the leading dimension of the array a . On exit, this value is unaltered.
n	On entry, this is the order of the matrix a . On exit, this value is unaltered.
info	On exit, this signals success: • = 0 for normal return. • = k signals an error condition. the leading minor of order k is not positive definite.

Definition at line 29 of file dpofa.f.

Here is the caller graph for this function:



3.11 driver1.f File Reference

Functions/Subroutines

- program **driver**

3.12 driver1.f90 File Reference

Functions/Subroutines

- program **driver**

3.13 driver2.f File Reference

Functions/Subroutines

- program **driver**

3.14 driver2.f90 File Reference

Functions/Subroutines

- program **driver**

3.15 driver3.f File Reference

Functions/Subroutines

- program **driver**

3.16 driver3.f90 File Reference

Functions/Subroutines

- program **driver**

3.17 dscal.f File Reference

Functions/Subroutines

- subroutine dscal (n, da, dx, incx)

  scales a vector by a constant.

3.17.1 Function/Subroutine Documentation

3.17.1.1 dscal()

```
subroutine dscal (
             integer n,
             double precision da,
             double precision, dimension(*) dx,
             integer incx )
```

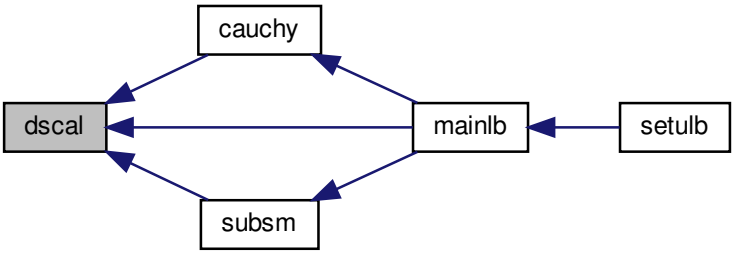
scales a vector by a constant. uses unrolled loops for increment equal to one.

Parameters

n	dimensionality of the vectors
da	scaling factor to be applied on dx
dx	vector to be scaled
incx	spacing of elements in dx

Definition at line 13 of file dscal.f.

Here is the caller graph for this function:



3.18 dtrsl.f File Reference

Functions/Subroutines

- subroutine dtrsl (t, ldt, n, b, job, info)

  dtrsl solves triangular systems.

3.18.1 Function/Subroutine Documentation

3.18.1.1 dtrsl()

```
subroutine dtrsl (
             double precision, dimension(ldt,*) t,
             integer ldt,
             integer n,
             double precision, dimension(*) b,
             integer job,
             integer info )
```

dtrsl solves systems of the form

```
t * x = b
```

or trans(t) ∗ x = b

where t is a triangular matrix of order n. here trans(t) denotes the transpose of the matrix t.

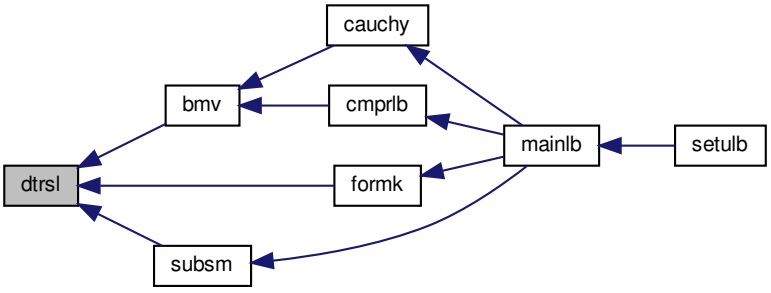
Parameters

t	On entry, t contains the matrix of the system. the zero elements of the matrix are not referenced, and the corresponding elements of the array can be used to store other information.
ldt	On entry, ldt is the leading dimension of the array t.
n	On entry, n is the order of the system.
b	On entry, b contains the right hand side of the system. On exit, b contains the solution, if info .eq. 0. otherwise b is unaltered.
job	On entry, job specifies what kind of system is to be solved. if job is • 00 solve t∗x=b, t lower triangular, • 01 solve t∗x=b, t upper triangular, • 10 solve trans(t)∗x=b, t lower triangular, • 11 solve trans(t)∗x=b, t upper triangular.
info	On exit, info contains zero if the system is nonsingular. otherwise info contains the index of the first zero diagonal element of t.

Definition at line 33 of file dtrsl.f.

Here is the call graph for this function:

Here is the caller graph for this function:



3.19 errclb.f File Reference

Functions/Subroutines

- subroutine errclb (n, m, factr, l, u, nbd, task, info, k)

  This subroutine checks the validity of the input data.

3.19.1 Function/Subroutine Documentation

3.19.1.1 errclb()

```
subroutine errclb (
             integer n,
             integer m,
             double precision factr,
             double precision, dimension(n) l,
             double precision, dimension(n) u,
             integer, dimension(n) nbd,
             character*60 task,
             integer info,
             integer k )
```

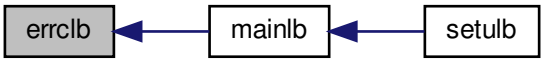
This subroutine checks the validity of the input data.

Parameters

n	number of parameters
m	history size of approximated Hessian
factr	convergence criterion on function value
l	lower bounds for parameters
u	upper bounds for parameters
nbd	indicates which bounds are present
task	what task is to be performed
info	error/success indicator
k	index of last errournous parameter

Definition at line 17 of file errclb.f.

Here is the caller graph for this function:



3.20 formk.f File Reference

Functions/Subroutines

- subroutine formk (n, nsub, ind, nenter, ileave, indx2, iupdat, updatd, wn, wn1, m, ws, wy, sy, theta, col, head, info)

  Forms the LEL^T factorization of the indefinite matrix K.

3.20.1 Function/Subroutine Documentation

3.20.1.1 formk()

```
subroutine formk (
             integer n,
             integer nsub,
             integer, dimension(n) ind,
             integer nenter,
             integer ileave,
             integer, dimension(n) indx2,
             integer iupdat,
             logical updatd,
             double precision, dimension(2*m, 2*m) wn,
             double precision, dimension(2*m, 2*m) wn1,
             integer m,
             double precision, dimension(n, m) ws,
             double precision, dimension(n, m) wy,
             double precision, dimension(m, m) sy,
             double precision theta,
             integer col,
             integer head,
             integer info )
```

This subroutine forms the LEL^T factorization of the indefinite matrix

K = [-D -Y'ZZ'Y/theta L_a'-R_z'] [L_a -R_z theta∗S'AA'S] where E = [-I 0] [0 I]

The matrix K can be shown to be equal to the matrix M^[-1]N occurring in section 5.1 of [1], as well as to the matrix Mbar^[-1] Nbar in section 5.3.

Parameters

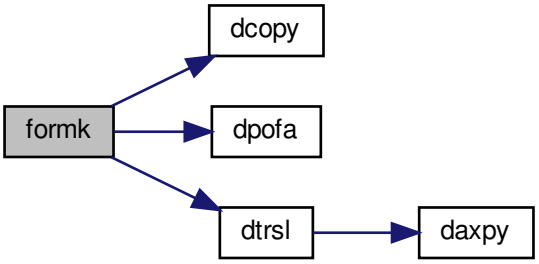
n	On entry n is the dimension of the problem. On exit n is unchanged.
nsub	On entry nsub is the number of subspace variables in free set. On exit nsub is not changed.
ind	On entry ind specifies the indices of subspace variables. On exit ind is unchanged.
nenter	On entry nenter is the number of variables entering the free set. On exit nenter is unchanged.
ileave	On entry indx2(ileave),...,indx2(n) are the variables leaving the free set. On exit ileave is unchanged.

Parameters

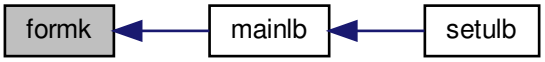
indx2	On entry indx2(1),...,indx2(nenter) are the variables entering the free set, while indx2(ileave),...,indx2(n) are the variables leaving the free set. On exit indx2 is unchanged.
iupdat	On entry iupdat is the total number of BFGS updates made so far. On exit iupdat is unchanged.
updatd	On entry 'updatd' is true if the L-BFGS matrix is updated. On exit 'updatd' is unchanged.
wn	On entry wn is unspecified. On exit the upper triangle of wn stores the LEL^T factorization of the 2∗col x 2∗col indefinite matrix [-D -Y'ZZ'Y/theta L_a'-R_z'] [L_a -R_z theta∗S'AA'S]
wn1	On entry wn1 stores the lower triangular part of [Y' ZZ'Y L_a'+R_z'] [L_a+R_z S'AA'S] in the previous iteration. On exit wn1 stores the corresponding updated matrices. The purpose of wn1 is just to store these inner products so they can be easily updated and inserted into wn.
m	On entry m is the maximum number of variable metric corrections used to define the limited memory matrix. On exit m is unchanged.
ws	On entry this stores S, a set of s-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wy	On entry this stores Y, a set of y-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
sy	On entry this stores S'Y, that defines the limited memory BFGS matrix. On exit this array is unchanged.
theta	On entry theta is the scaling factor specifying B_0 = theta I. On exit theta is unchanged.
col	On entry col is the actual number of variable metric corrections stored so far. On exit col is unchanged.
head	On entry head is the location of the first s-vector (or y-vector) in S (or Y). On exit col is unchanged.
info	On entry info is unspecified. On exit info • = 0 for normal return; • = -1 when the 1st Cholesky factorization failed; • = -2 when the 2st Cholesky factorization failed.

Definition at line 92 of file formk.f.

Here is the call graph for this function:



Here is the caller graph for this function:



3.21 formt.f File Reference

Functions/Subroutines

- subroutine formt (m, wt, sy, ss, col, theta, info)

  Forms the upper half of the pos. def. and symm. T.

3.21.1 Function/Subroutine Documentation

3.21.1.1 formt()

```
subroutine formt (
             integer m,
             double precision, dimension(m, m) wt,
             double precision, dimension(m, m) sy,
             double precision, dimension(m, m) ss,
             integer col,
             double precision theta,
             integer info )
```

This subroutine forms the upper half of the pos. def. and symm. T = theta∗SS + L∗D^(-1)∗L', stores T in the upper triangle of the array wt, and performs the Cholesky factorization of T to produce J∗J', with J' stored in the upper triangle of wt.

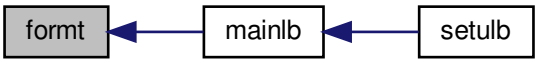
Parameters

m	history size of approximated Hessian
wt	part of L-BFGS matrix
sy	part of L-BFGS matrix
ss	part of L-BFGS matrix
col	On entry col is the actual number of variable metric corrections stored so far. On exit col is unchanged.
theta	On entry theta is the scaling factor specifying B_0 = theta I. On exit theta is unchanged.
info	error/success indicator

Definition at line 22 of file formt.f.

Here is the call graph for this function:

Here is the caller graph for this function:



3.22 freev.f File Reference

Functions/Subroutines

- subroutine freev (n, nfree, index, nenter, ileave, indx2, iwhere, wrk, updatd, cnstnd, iprint, iter)

  This subroutine counts the entering and leaving variables when iter > 0, and finds the index set of free and active variables at the GCP.

3.22.1 Function/Subroutine Documentation

3.22.1.1 freev()

```
subroutine freev (
             integer n,
             integer nfree,
             integer, dimension(n) index,
             integer nenter,
             integer ileave,
             integer, dimension(n) indx2,
             integer, dimension(n) iwhere,
             logical wrk,
             logical updatd,
             logical cnstnd,
             integer iprint,
             integer iter )
```

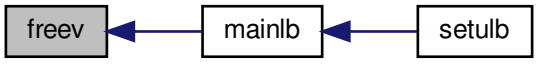
This subroutine counts the entering and leaving variables when iter $>$ 0, and finds the index set of free and active variables at the GCP.

Parameters

n	number of parameters
nfree	number of free parameters, i.e., those not at their bounds
index	for i=1,...,nfree, index(i) are the indices of free variables for i=nfree+1,...,n, index(i) are the indices of bound variables On entry after the first iteration, index gives the free variables at the previous iteration. On exit it gives the free variables based on the determination in cauchy using the array iwhere.
nenter	TODO
ileave	TODO
indx2	On entry indx2 is unspecified. On exit with iter$>$0, indx2 indicates which variables have changed status since the previous iteration. For i= 1,...,nenter, indx2(i) have changed from bound to free. For i= ileave+1,...,n, indx2(i) have changed from free to bound.
iwhere	TODO
wrk	TODO
updatd	TODO
cnstnd	indicating whether bounds are present
iprint	control screen output
iter	TODO

Definition at line 34 of file freev.f.

Here is the caller graph for this function:



3.23 hpsolb.f File Reference

Functions/Subroutines

- subroutine hpsolb (n, t, iorder, iheap)

  This subroutine sorts out the least element of t, and puts the remaining elements of t in a heap.

3.23.1 Function/Subroutine Documentation

3.23.1.1 hpsolb()

```
subroutine hpsolb (
             integer n,
             double precision, dimension(n) t,
             integer, dimension(n) iorder,
             integer iheap )
```

Parameters

n	On entry n is the dimension of the arrays t and iorder. On exit n is unchanged.
t	On entry t stores the elements to be sorted. On exit t(n) stores the least elements of t, and t(1) to t(n-1) stores the remaining elements in the form of a heap.
iorder	On entry iorder(i) is the index of t(i). On exit iorder(i) is still the index of t(i), but iorder may be permuted in accordance with t.
iheap	On entry iheap should be set as follows: • iheap .eq. 0 if t(1) to t(n) is not in the form of a heap, • iheap .ne. 0 if otherwise. On exit iheap is unchanged.

Definition at line 22 of file hpsolb.f.

Here is the caller graph for this function:

3.24 lnsrlb.f File Reference

Functions/Subroutines

- subroutine lnsrlb (n, l, u, nbd, x, f, fold, gd, gdold, g, d, r, t, z, stp, dnorm, dtd, xstep, stpmx, iter, ifun, iback, nfgv, info, task, boxed, cnstnd, csave, isave, dsave)

  This subroutine calls subroutine dcsrch from the Minpack2 library to perform the line search. Subroutine dscrch is safeguarded so that all trial points lie within the feasible region.

3.24.1 Function/Subroutine Documentation

3.24.1.1 lnsrlb()

```
subroutine lnsrlb (
            integer n,
            double precision, dimension(n) l,
            double precision, dimension(n) u,
            integer, dimension(n) nbd,
            double precision, dimension(n) x,
            double precision f,
            double precision fold,
            double precision gd,
            double precision gdold,
            double precision, dimension(n) g,
            double precision, dimension(n) d,
            double precision, dimension(n) r,
            double precision, dimension(n) t,
            double precision, dimension(n) z,
            double precision stp,
            double precision dnorm,
            double precision dtd,
            double precision xstep,
            double precision stpmx,
            integer iter,
            integer ifun,
            integer iback,
            integer nfgv,
            integer info,
            character*60 task,
            logical boxed,
            logical cnstnd,
            character*60 csave,
            integer, dimension(2) isave,
            double precision, dimension(13) dsave )
```

Parameters

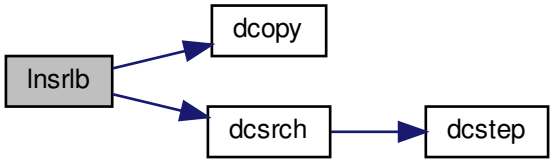
n	number of parameters
l	lower bounds of parameters
u	upper bounds of parameters

Parameters

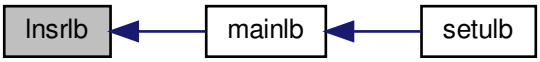
nbd	On entry nbd represents the type of bounds imposed on the variables, and must be specified as follows: nbd(i)= • 0 if x(i) is unbounded, • 1 if x(i) has only a lower bound, • 2 if x(i) has both lower and upper bounds, and • 3 if x(i) has only an upper bound. On exit nbd is unchanged.
x	position
f	function value at x
fold	TODO
gd	TODO
gdold	TODO
g	gradient of f at x
d	TODO
r	TODO
t	TODO
z	TODO
stp	TODO
dnorm	TODO
dtd	TODO
xstep	TODO
stpmx	TODO
iter	TODO
ifun	TODO
iback	TODO
nfgv	TODO
info	TODO
task	TODO
boxed	TODO
cnstnd	TODO
csave	working array
isave	working array
dsave	working array

Definition at line 47 of file lnsrlb.f.

Here is the call graph for this function:



Here is the caller graph for this function:



3.25 mainlb.f File Reference

Functions/Subroutines

- subroutine mainlb (n, m, x, l, u, nbd, f, g, factr, pgtol, ws, wy, sy, ss, wt, wn, snd, z, r, d, t, xp, wa, index, iwhere, indx2, task, iprint, csave, lsave, isave, dsave)

  This subroutine solves bound constrained optimization problems by using the compact formula of the limited memory BFGS updates.

3.25.1 Function/Subroutine Documentation

3.25.1.1 mainlb()

```
subroutine mainlb (
            integer n,
            integer m,
            double precision, dimension(n) x,
            double precision, dimension(n) l,
            double precision, dimension(n) u,
            integer, dimension(n) nbd,
            double precision f,
            double precision, dimension(n) g,
            double precision factr,
```

```
double precision pgtol,
double precision, dimension(n, m) ws,
double precision, dimension(n, m) wy,
double precision, dimension(m, m) sy,
double precision, dimension(m, m) ss,
double precision, dimension(m, m) wt,
double precision, dimension(2*m, 2*m) wn,
double precision, dimension(2*m, 2*m) snd,
double precision, dimension(n) z,
double precision, dimension(n) r,
double precision, dimension(n) d,
double precision, dimension(n) t,
double precision, dimension(n) xp,
double precision, dimension(8*m) wa,
integer, dimension(n) index,
integer, dimension(n) iwhere,
integer, dimension(n) indx2,
character*60 task,
integer iprint,
character*60 csave,
logical, dimension(4) lsave,
integer, dimension(23) isave,
double precision, dimension(29) dsave )
```

This subroutine solves bound constrained optimization problems by using the compact formula of the limited memory BFGS updates.

Parameters

n	On entry n is the number of variables. On exit n is unchanged.
m	On entry m is the maximum number of variable metric corrections allowed in the limited memory matrix. On exit m is unchanged.
x	On entry x is an approximation to the solution. On exit x is the current approximation.
l	On entry l is the lower bound of x. On exit l is unchanged.
u	On entry u is the upper bound of x. On exit u is unchanged.
nbd	On entry nbd represents the type of bounds imposed on the variables, and must be specified as follows: nbd(i)= • 0 if x(i) is unbounded, • 1 if x(i) has only a lower bound, • 2 if x(i) has both lower and upper bounds, • 3 if x(i) has only an upper bound. On exit nbd is unchanged.
f	On first entry f is unspecified. On final exit f is the value of the function at x.
g	On first entry g is unspecified. On final exit g is the value of the gradient at x.

Parameters

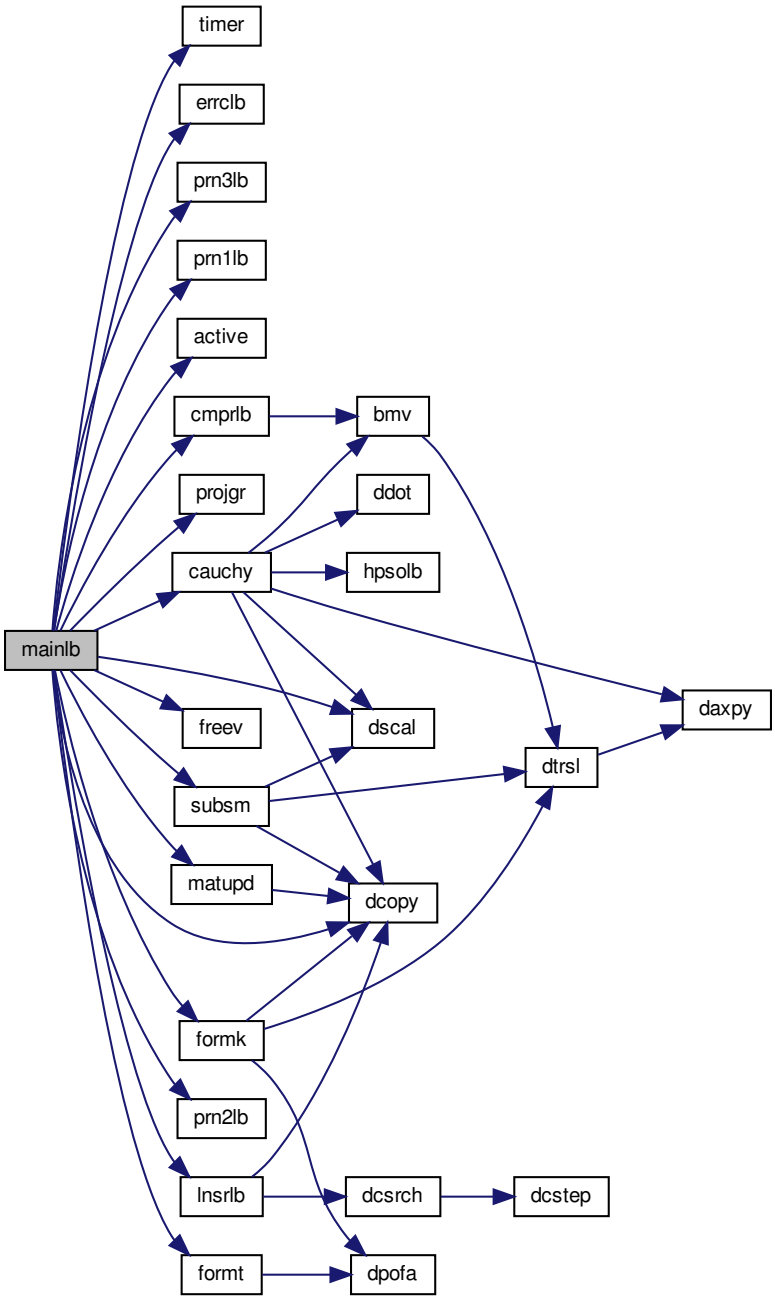
factr	On entry factr >= 0 is specified by the user. The iteration will stop when (f^k - f^{k+1})/max{|f^k|,|f^{k+1}|,1} <= factr∗epsmch where epsmch is the machine precision, which is automatically generated by the code. On exit factr is unchanged.
pgtol	On entry pgtol >= 0 is specified by the user. The iteration will stop when max{|proj g_i | i = 1, ..., n} <= pgtol where pg_i is the ith component of the projected gradient. On exit pgtol is unchanged.
ws	On entry this stores S, a set of s-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wy	On entry this stores Y, a set of y-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
sy	On entry this stores S'Y, that defines the limited memory BFGS matrix. On exit this array is unchanged.
ss	On entry this stores S'S, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wt	On entry this stores the Cholesky factorization of (theta∗S'S+LD^(-1)L'), that defines the limited memory BFGS matrix. See eq. (2.26) in [3]. On exit this array is unchanged.
wn	working array used to store the LEL^T factorization of the indefinite matrix K = [-D -Y'ZZ'Y/theta L_a'-R_z'] [L_a -R_z theta∗S'AA'S] where E = [-I 0] [0 I]
snd	working array used to store the lower triangular part of N = [Y' ZZ'Y L_a'+R_z'] [L_a +R_z S'AA'S]
z	working array used at different times to store the Cauchy point and the Newton point.
r	working array
d	working array
t	working array
xp	working array used to safeguard the projected Newton direction
wa	working array
index	In subroutine freev, index is used to store the free and fixed variables at the Generalized Cauchy Point (GCP).
iwhere	working array used to record the status of the vector x for GCP computation. iwhere(i)= • 0 or -3 if x(i) is free and has bounds, • 1 if x(i) is fixed at l(i), and l(i) .ne. u(i) • 2 if x(i) is fixed at u(i), and u(i) .ne. l(i) • 3 if x(i) is always fixed, i.e., u(i)=x(i)=l(i) • -1 if x(i) is always free, i.e., no bounds on it.
indx2	working array Within subroutine cauchy, indx2 corresponds to the array iorder. In subroutine freev, a list of variables entering and leaving the free set is stored in indx2, and it is passed on to subroutine formk with this information.
task	working string indicating the current job when entering and leaving this subroutine.

Parameters

iprint	It controls the frequency and type of output generated: • iprint<0 no output is generated; • iprint=0 print only one line at the last iteration; • 0<iprint<99 print also f and |proj g| every iprint iterations; • iprint=99 print details of every iteration except n-vectors; • iprint=100 print also the changes of active set and final x; • iprint>100 print details of every iteration including x and g; When iprint > 0, the file iterate.dat will be created to summarize the iteration.
csave	working string
lsave	working array
isave	working array
dsave	working array

Definition at line 132 of file mainlb.f.

Here is the call graph for this function:



Here is the caller graph for this function:

3.26 matupd.f File Reference

Functions/Subroutines

- subroutine matupd (n, m, ws, wy, sy, ss, d, r, itail, iupdat, col, head, theta, rr, dr, stp, dtd)

  This subroutine updates matrices WS and WY, and forms the middle matrix in B.

3.26.1 Function/Subroutine Documentation

3.26.1.1 matupd()

```
subroutine matupd (
             integer n,
             integer m,
             double precision, dimension(n, m) ws,
             double precision, dimension(n, m) wy,
             double precision, dimension(m, m) sy,
             double precision, dimension(m, m) ss,
             double precision, dimension(n) d,
             double precision, dimension(n) r,
             integer itail,
             integer iupdat,
             integer col,
             integer head,
             double precision theta,
             double precision rr,
             double precision dr,
             double precision stp,
             double precision dtd )
```

This subroutine updates matrices WS and WY, and forms the middle matrix in B.

Parameters

n	On entry n is the number of variables. On exit n is unchanged.
m	On entry m is the maximum number of variable metric corrections allowed in the limited memory matrix. On exit m is unchanged.

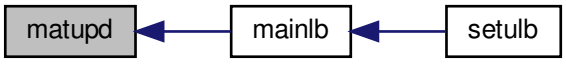
Parameters

ws	On entry this stores S, a set of s-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wy	On entry this stores Y, a set of y-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
sy	On entry this stores S'Y, that defines the limited memory BFGS matrix. On exit this array is unchanged.
ss	On entry this stores S'S, that defines the limited memory BFGS matrix. On exit this array is unchanged.
d	TODO
r	TODO
itail	TODO
iupdat	TODO
col	On entry col is the actual number of variable metric corrections stored so far. On exit col is unchanged.
head	On entry head is the location of the first s-vector (or y-vector) in S (or Y). On exit col is unchanged.
theta	On entry theta is the scaling factor specifying B_0 = theta I. On exit theta is unchanged.
rr	TODO
dr	TODO
stp	TODO
dtd	TODO

Definition at line 51 of file matupd.f.

Here is the call graph for this function:

Here is the caller graph for this function:



3.27 prn1lb.f File Reference

Functions/Subroutines

- subroutine prn1lb (n, m, l, u, x, iprint, itfile, epsmch)

  This subroutine prints the input data, initial point, upper and lower bounds of each variable, machine precision, as well as the headings of the output.

3.27.1 Function/Subroutine Documentation

3.27.1.1 prn1lb()

```
subroutine prn1lb (
            integer n,
            integer m,
            double precision, dimension(n) l,
            double precision, dimension(n) u,
            double precision, dimension(n) x,
            integer iprint,
            integer itfile,
            double precision epsmch )
```

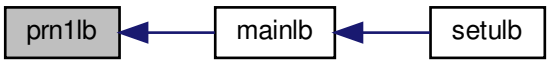
This subroutine prints the input data, initial point, upper and lower bounds of each variable, machine precision, as well as the headings of the output.

Parameters

n	On entry n is the number of variables. On exit n is unchanged.
m	On entry m is the maximum number of variable metric corrections allowed in the limited memory matrix. On exit m is unchanged.
l	On entry l is the lower bound of x. On exit l is unchanged.
u	On entry u is the upper bound of x. On exit u is unchanged.
x	On entry x is an approximation to the solution. On exit x is the current approximation.
iprint	It controls the frequency and type of output generated: • iprint$<$0 no output is generated; • iprint=0 print only one line at the last iteration; • 0$<$iprint$<$99 print also f and $\|$proj g$\|$ every iprint iterations; • iprint=99 print details of every iteration except n-vectors; • iprint=100 print also the changes of active set and final x; • iprint$>$100 print details of every iteration including x and g; When iprint $>$ 0, the file iterate.dat will be created to summarize the iteration.
itfile	unit number of iterate.dat file
epsmch	machine precision epsilon

Definition at line 40 of file prn1lb.f.

Here is the caller graph for this function:



3.28 prn2lb.f File Reference

Functions/Subroutines

- subroutine prn2lb (n, x, f, g, iprint, itfile, iter, nfgv, nact, sbgnrm, nseg, word, iword, iback, stp, xstep)
  This subroutine prints out new information after a successful line search.

3.28.1 Function/Subroutine Documentation

3.28.1.1 prn2lb()

```
subroutine prn2lb (
             integer n,
             double precision, dimension(n) x,
             double precision f,
             double precision, dimension(n) g,
             integer iprint,
             integer itfile,
             integer iter,
             integer nfgv,
             integer nact,
             double precision sbgnrm,
             integer nseg,
             character*3 word,
             integer iword,
             integer iback,
             double precision stp,
             double precision xstep )
```

This subroutine prints out new information after a successful line search.

Parameters

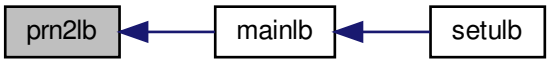
n	On entry n is the number of variables. On exit n is unchanged.
x	On entry x is an approximation to the solution. On exit x is the current approximation.

Parameters

f	On first entry f is unspecified. On final exit f is the value of the function at x.
g	On first entry g is unspecified. On final exit g is the value of the gradient at x.
iprint	It controls the frequency and type of output generated: • iprint<0 no output is generated; • iprint=0 print only one line at the last iteration; • 0<iprint<99 print also f and |proj g| every iprint iterations; • iprint=99 print details of every iteration except n-vectors; • iprint=100 print also the changes of active set and final x; • iprint>100 print details of every iteration including x and g; When iprint > 0, the file iterate.dat will be created to summarize the iteration.
itfile	unit number of iterate.dat file
iter	TODO
nfgv	TODO
nact	TODO
sbgnrm	TODO
nseg	TODO
word	TODO
iword	TODO
iback	TODO
stp	TODO
xstep	TODO

Definition at line 45 of file prn2lb.f.

Here is the caller graph for this function:



3.29 prn3lb.f File Reference

Functions/Subroutines

- subroutine prn3lb (n, x, f, task, iprint, info, itfile, iter, nfgv, nintol, nskip, nact, sbgnrm, time, nseg, word, iback, stp, xstep, k, cachyt, sbtime, lnscht)

  This subroutine prints out information when either a built-in convergence test is satisfied or when an error message is generated.

3.29.1 Function/Subroutine Documentation

3.29.1.1 prn3lb()

```
subroutine prn3lb (
             integer n,
             double precision, dimension(n) x,
             double precision f,
             character*60 task,
             integer iprint,
             integer info,
             integer itfile,
             integer iter,
             integer nfgv,
             integer nintol,
             integer nskip,
             integer nact,
             double precision sbgnrm,
             double precision time,
             integer nseg,
             character*3 word,
             integer iback,
             double precision stp,
             double precision xstep,
             integer k,
             double precision cachyt,
             double precision sbtime,
             double precision lnscht )
```

This subroutine prints out information when either a built-in convergence test is satisfied or when an error message is generated.

Parameters

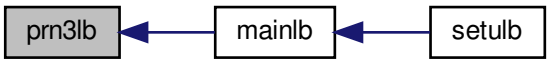
n	On entry n is the number of variables. On exit n is unchanged.
x	On entry x is an approximation to the solution. On exit x is the current approximation.
f	On first entry f is unspecified. On final exit f is the value of the function at x.
task	working string indicating the current job when entering and leaving this subroutine.
iprint	TODO
info	TODO
itfile	unit number of iterate.dat file
iter	TODO
nfgv	TODO
nintol	TODO
nskip	TODO
nact	TODO
sbgnrm	TODO
time	TODO
nseg	TODO
word	TODO

Parameters

iback	TODO
stp	TODO
xstep	TODO
k	TODO
cachyt	TODO
sbtime	TODO
lnscht	TODO

Definition at line 45 of file prn3lb.f.

Here is the caller graph for this function:



3.30 projgr.f File Reference

Functions/Subroutines

- subroutine projgr (n, l, u, nbd, x, g, sbgnrm)

  This subroutine computes the infinity norm of the projected gradient.

3.30.1 Function/Subroutine Documentation

3.30.1.1 projgr()

```
subroutine projgr (
             integer n,
             double precision, dimension(n) l,
             double precision, dimension(n) u,
             integer, dimension(n) nbd,
             double precision, dimension(n) x,
             double precision, dimension(n) g,
             double precision sbgnrm )
```

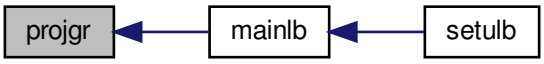
This subroutine computes the infinity norm of the projected gradient.

Parameters

n	On entry n is the number of variables. On exit n is unchanged.
l	On entry l is the lower bound of x. On exit l is unchanged.
u	On entry u is the upper bound of x. On exit u is unchanged.
nbd	On entry nbd represents the type of bounds imposed on the variables, and must be specified as follows: nbd(i)= • 0 if x(i) is unbounded, • 1 if x(i) has only a lower bound, • 2 if x(i) has both lower and upper bounds, • 3 if x(i) has only an upper bound. On exit nbd is unchanged.
x	On entry x is an approximation to the solution. On exit x is unchanged.
g	On entry g is the gradient. On exit g is unchanged.
sbgnrm	infinity norm of projected gradient

Definition at line 34 of file projgr.f.

Here is the caller graph for this function:



3.31 setulb.f File Reference

Functions/Subroutines

- subroutine setulb (n, m, x, l, u, nbd, f, g, factr, pgtol, wa, iwa, task, iprint, csave, lsave, isave, dsave)

  This subroutine partitions the working arrays wa and iwa, and then uses the limited memory BFGS method to solve the bound constrained optimization problem by calling mainlb.

3.31.1 Function/Subroutine Documentation

3.31.1.1 setulb()

```
subroutine setulb (
             integer n,
             integer m,
             double precision, dimension(n) x,
             double precision, dimension(n) l,
             double precision, dimension(n) u,
             integer, dimension(n) nbd,
             double precision f,
             double precision, dimension(n) g,
             double precision factr,
             double precision pgtol,
             double precision, dimension(2*m*n + 5*n + 11*m*m + 8*m) wa,
             integer, dimension(3*n) iwa,
             character*60 task,
             integer iprint,
             character*60 csave,
             logical, dimension(4) lsave,
             integer, dimension(44) isave,
             double precision, dimension(29) dsave )
```

This subroutine partitions the working arrays wa and iwa, and then uses the limited memory BFGS method to solve the bound constrained optimization problem by calling mainlb. (The direct method will be used in the subspace minimization.)

Parameters

n	On entry n is the dimension of the problem. On exit n is unchanged.
m	On entry m is the maximum number of variable metric corrections used to define the limited memory matrix. On exit m is unchanged.
x	On entry x is an approximation to the solution. On exit x is the current approximation.
l	On entry l is the lower bound on x. On exit l is unchanged.
u	On entry u is the upper bound on x. On exit u is unchanged.
nbd	On entry nbd represents the type of bounds imposed on the variables, and must be specified as follows: nbd(i)= • 0 if x(i) is unbounded, • 1 if x(i) has only a lower bound, • 2 if x(i) has both lower and upper bounds, and • 3 if x(i) has only an upper bound. On exit nbd is unchanged.
f	On first entry f is unspecified. On final exit f is the value of the function at x.
g	On first entry g is unspecified. On final exit g is the value of the gradient at x.

Parameters

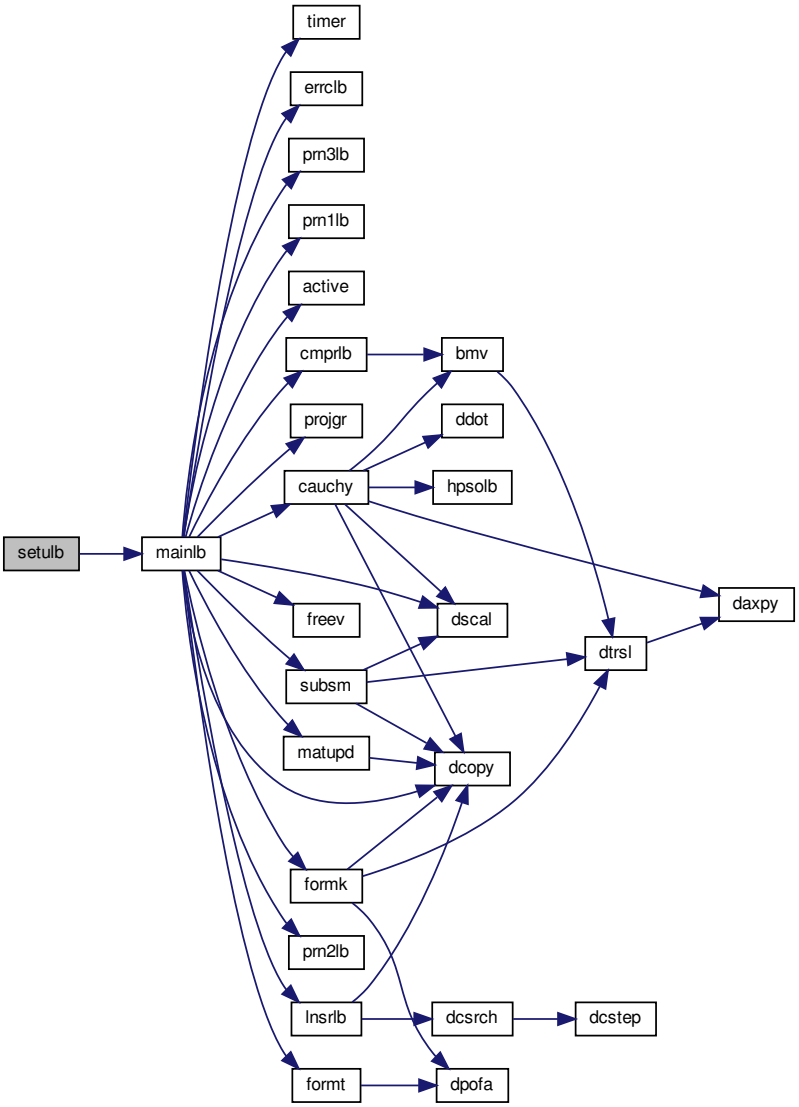
factr	On entry factr >= 0 is specified by the user. The iteration will stop when (f^k - f^{k+1})/max{|f^k|,|f^{k+1}|,1} <= factr∗epsmch where epsmch is the machine precision, which is automatically generated by the code. Typical values for factr: • 1.d+12 for low accuracy; • 1.d+7 for moderate accuracy; • 1.d+1 for extremely high accuracy. On exit factr is unchanged.
pgtol	On entry pgtol >= 0 is specified by the user. The iteration will stop when max{|proj g_i | i = 1, ..., n} <= pgtol where pg_i is the ith component of the projected gradient. On exit pgtol is unchanged.
wa	working array
iwa	working array
task	working string indicating the current job when entering and quitting this subroutine.
iprint	Must be set by the user. It controls the frequency and type of output generated: • iprint<0 no output is generated; • iprint=0 print only one line at the last iteration; • 0<iprint<99 print also f and |proj g| every iprint iterations; • iprint=99 print details of every iteration except n-vectors; • iprint=100 print also the changes of active set and final x; • iprint>100 print details of every iteration including x and g; When iprint > 0, the file iterate.dat will be created to summarize the iteration.
csave	working string
lsave	working array; On exit with 'task' = NEW_X, the following information is available: • If lsave(1) = .true. then the initial X has been replaced by its projection in the feasible set; • If lsave(2) = .true. then the problem is constrained; • If lsave(3) = .true. then each variable has upper and lower bounds;

Parameters

isave	working array; On exit with 'task' = NEW_X, the following information is available: • isave(22) = the total number of intervals explored in the search of Cauchy points; • isave(26) = the total number of skipped BFGS updates before the current iteration; • isave(30) = the number of current iteration; • isave(31) = the total number of BFGS updates prior the current iteration; • isave(33) = the number of intervals explored in the search of Cauchy point in the current iteration; • isave(34) = the total number of function and gradient evaluations; • isave(36) = the number of function value or gradient evaluations in the current iteration; • if isave(37) = 0 then the subspace argmin is within the box; • if isave(37) = 1 then the subspace argmin is beyond the box; • isave(38) = the number of free variables in the current iteration; • isave(39) = the number of active constraints in the current iteration; • n + 1 - isave(40) = the number of variables leaving the set of active constraints in the current iteration; • isave(41) = the number of variables entering the set of active constraints in the current iteration.
dsave	working array; On exit with 'task' = NEW_X, the following information is available: • dsave(1) = current 'theta' in the BFGS matrix; • dsave(2) = f(x) in the previous iteration; • dsave(3) = factr∗epsmch; • dsave(4) = 2-norm of the line search direction vector; • dsave(5) = the machine precision epsmch generated by the code; • dsave(7) = the accumulated time spent on searching for Cauchy points; • dsave(8) = the accumulated time spent on subspace minimization; • dsave(9) = the accumulated time spent on line search; • dsave(11) = the slope of the line search function at the current point of line search; • dsave(12) = the maximum relative step length imposed in line search; • dsave(13) = the infinity norm of the projected gradient; • dsave(14) = the relative step length in the line search; • dsave(15) = the slope of the line search function at the starting point of the line search; • dsave(16) = the square of the 2-norm of the line search direction vector.

Definition at line 140 of file setulb.f.

Here is the call graph for this function:



3.32 subsm.f File Reference

Functions/Subroutines

- subroutine subsm (n, m, nsub, ind, l, u, nbd, x, d, xp, ws, wy, theta, xx, gg, col, head, iword, wv, wn, iprint, info)

  Performs the subspace minimization.

3.32.1 Function/Subroutine Documentation

3.32.1.1 subsm()

```
subroutine subsm (
             integer n,
             integer m,
             integer nsub,
             integer, dimension(nsub) ind,
             double precision, dimension(n) l,
             double precision, dimension(n) u,
             integer, dimension(n) nbd,
             double precision, dimension(n) x,
             double precision, dimension(n) d,
             double precision, dimension(n) xp,
             double precision, dimension(n, m) ws,
             double precision, dimension(n, m) wy,
             double precision theta,
             double precision, dimension(n) xx,
             double precision, dimension(n) gg,
             integer col,
             integer head,
             integer iword,
             double precision, dimension(2*m) wv,
             double precision, dimension(2*m, 2*m) wn,
             integer iprint,
             integer info )
```

Given xcp, l, u, r, an index set that specifies the active set at xcp, and an l-BFGS matrix B (in terms of WY, WS, SY, WT, head, col, and theta), this subroutine computes an approximate solution of the subspace problem

(P) min Q(x) = r'(x-xcp) + 1/2 (x-xcp)' B (x-xcp)

subject to l<=x<=u x_i=xcp_i for all i in A(xcp)

along the subspace unconstrained Newton direction

d = -(Z'BZ)^(-1) r.

The formula for the Newton direction, given the L-BFGS matrix and the Sherman-Morrison formula, is

d = (1/theta)r + (1/theta*2) Z'WK^(-1)W'Z r.

where K = [-D -Y'ZZ'Y/theta L_a'-R_z'] [L_a -R_z theta*S'AA'S]

Note that this procedure for computing d differs from that described in [1]. One can show that the matrix K is equal to the matrix M^[-1]N in that paper.

Parameters

n	On entry n is the dimension of the problem. On exit n is unchanged.
m	On entry m is the maximum number of variable metric corrections used to define the limited memory matrix. On exit m is unchanged.
nsub	On entry nsub is the number of free variables. On exit nsub is unchanged.
ind	On entry ind specifies the coordinate indices of free variables. On exit ind is unchanged.

Parameters

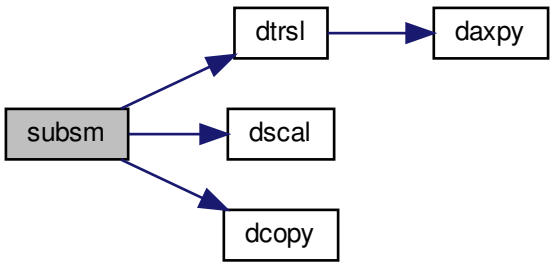
l	On entry l is the lower bound of x. On exit l is unchanged.
u	On entry u is the upper bound of x. On exit u is unchanged.
nbd	On entry nbd represents the type of bounds imposed on the variables, and must be specified as follows: nbd(i)= • 0 if x(i) is unbounded, • 1 if x(i) has only a lower bound, • 2 if x(i) has both lower and upper bounds, and • 3 if x(i) has only an upper bound. On exit nbd is unchanged.
x	On entry x specifies the Cauchy point xcp. On exit x(i) is the minimizer of Q over the subspace of free variables.
d	On entry d is the reduced gradient of Q at xcp. On exit d is the Newton direction of Q.
xp	used to safeguard the projected Newton direction
xx	On entry it holds the current iterate. On output it is unchanged.
gg	On entry it holds the gradient at the current iterate. On output it is unchanged.
ws	On entry this stores S, a set of s-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
wy	On entry this stores Y, a set of y-vectors, that defines the limited memory BFGS matrix. On exit this array is unchanged.
theta	On entry theta is the scaling factor specifying B_0 = theta I. On exit theta is unchanged.
col	On entry col is the actual number of variable metric corrections stored so far. On exit col is unchanged.
head	On entry head is the location of the first s-vector (or y-vector) in S (or Y). On exit col is unchanged.
iword	On entry iword is unspecified. On exit iword specifies the status of the subspace solution. iword = • 0 if the solution is in the box, • 1 if some bound is encountered.
wv	working array
wn	On entry the upper triangle of wn stores the LEL^T factorization of the indefinite matrix K = [-D -Y'ZZ'Y/theta L_a'-R_z'] [L_a -R_z theta∗S'AA'S] where E = [-I 0] [0 I] On exit wn is unchanged.

Parameters

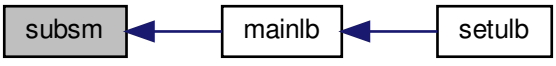
iprint	must be set by the user; It controls the frequency and type of output generated: • iprint<0 no output is generated; • iprint=0 print only one line at the last iteration; • 0<iprint<99 print also f and |proj g| every iprint iterations; • iprint=99 print details of every iteration except n-vectors; • iprint=100 print also the changes of active set and final x; • iprint>100 print details of every iteration including x and g; When iprint > 0, the file iterate.dat will be created to summarize the iteration.
info	On entry info is unspecified. On exit info = • 0 for normal return, • nonzero for abnormal return when the matrix K is ill-conditioned.

Definition at line 127 of file subsm.f.

Here is the call graph for this function:



Here is the caller graph for this function:



3.33 timer.f File Reference

Functions/Subroutines

- subroutine timer (ttime)

  This routine computes cpu time in double precision.

3.33.1 Function/Subroutine Documentation

3.33.1.1 timer()

```
subroutine timer (
             double precision ttime )
```

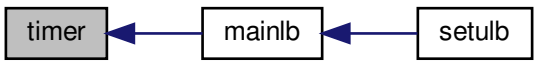
This routine computes cpu time in double precision; it makes use of the intrinsic f90 cpu_time therefore a conversion type is needed.

Parameters

ttime	CPU time in double precision

Definition at line 11 of file timer.f.

Here is the caller graph for this function:



Index