NAG Fortran Library Routine Document

E04LBF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

E04LBF is a comprehensive modified Newton algorithm for finding:

an unconstrained minimum of a function of several variables

a minimum of a function of several variables subject to fixed upper and/or lower bounds on the variables.

First and second derivatives are required. The routine is intended for functions which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

2 Specification

```
 SUBROUTINE E04LBF(N, FUNCT, HESS, MONIT, IPRINT, MAXCAL, ETA, XTOL,
1                  STEPMX, IBOUND, BL, BU, X, HESL, LH, HESD, ISTATE, F,
2                  G, IW, LIW, W, LW, IFAIL)
 INTEGER           N, IPRINT, MAXCAL, IBOUND, LH, ISTATE(N), IW(LIW),
1                  LIW, LW, IFAIL
 real              ETA, XTOL, STEPMX, BL(N), BU(N), X(N), HESL(LH),
1                  HESD(N), F, G(N), W(LW)
 EXTERNAL          FUNCT, HESS, MONIT
```

3 Description

This routine is applicable to problems of the form:

$$\text{Minimize } F(x_1, x_2, \ldots, x_n) \text{ subject to } l_j \leq x_j \leq u_j, \quad j = 1, 2, \ldots, n.$$

Special provision is made for unconstrained minimization (i.e., problems which actually have no bounds on the x_j), problems which have only non-negativity bounds, and problems in which $l_1 = l_2 = \cdots = l_n$ and $u_1 = u_2 = \cdots = u_n$. It is possible to specify that a particular x_j should be held constant. The user must supply a starting point, a subroutine FUNCT to calculate the value of $F(x)$ and its first derivatives $\dfrac{\partial F}{\partial x_j}$ at any point x, and a subroutine HESS to calculate the second derivatives $\dfrac{\partial^2 F}{\partial x_i \partial x_j}$.

A typical iteration starts at the current point x where n_z (say) variables are free from both their bounds. The vector of first derivatives of $F(x)$ with respect to the free variables, g_z, and the matrix of second derivatives with respect to the free variables, H, are obtained. (These both have dimension n_z.)

The equations

$$(H + E)p_z = -g_z$$

are solved to give a search direction p_z. (The matrix E is chosen so that $H + E$ is positive-definite.)

p_z is then expanded to an n-vector p by the insertion of appropriate zero elements; α is found such that $F(x + \alpha p)$ is approximately a minimum (subject to the fixed bounds) with respect to α, and x is replaced by $x + \alpha p$. (If a saddle point is found, a special search is carried out so as to move away from the saddle point.)

If any variable actually reaches a bound, it is fixed and n_z is reduced for the next iteration.

There are two sets of convergence criteria – a weaker and a stronger. Whenever the weaker criteria are satisfied, the Lagrange-multipliers are estimated for all active constraints. If any Lagrange-multiplier

estimate is significantly negative, then one of the variables associated with a negative Lagrange-multiplier estimate is released from its bound and the next search direction is computed in the extended subspace (i.e., n_z is increased). Otherwise, minimization continues in the current subspace until the stronger criteria are satisfied. If at this point there are no negative or near-zero Lagrange-multiplier estimates, the process is terminated.

If the user specifies that the problem is unconstrained, E04LBF sets the l_j to -10^6 and the u_j to 10^6. Thus, provided that the problem has been sensibly scaled, no bounds will be encountered during the minimization process and E04LBF will act as an unconstrained minimization algorithm.

4 References

Gill P E and Murray W (1973) Safeguarded steplength algorithms for optimization using descent methods *NPL Report NAC 37* National Physical Laboratory

Gill P E and Murray W (1974) Newton-type methods for unconstrained and linearly constrained optimization *Math. Program.* **7** 311–350

Gill P E and Murray W (1976) Minimization subject to bounds on the variables *NPL Report NAC 72* National Physical Laboratory

5 Parameters

1: N – INTEGER *Input*

On entry: the number n of independent variables.

Constraint: N ≥ 1.

2: FUNCT – SUBROUTINE, supplied by the user. *External Procedure*

FUNCT must evaluate the function $F(x)$ and its first derivatives $\dfrac{\partial F}{\partial x_j}$ at any point x. (However, if the user does not wish to calculate $F(x)$ or its first derivatives at a particular x, there is the option of setting a parameter to cause E04LBF to terminate immediately.)

Its specification is:

```
SUBROUTINE FUNCT(IFLAG, N, XC, FC, GC, IW, LIW, W, LW)
INTEGER          IFLAG, N, IW(LIW), LIW, LW
real             XC(N), FC, GC(N), W(LW)
```

1: IFLAG – INTEGER *Input/Output*

On entry: IFLAG will have been set to 2.

On exit: if it is not possible to evaluate $F(x)$ or its first derivatives at the point x given in XC (or if it is wished to stop the calculation for any other reason) the user should reset IFLAG to some negative number and return control to E04LBF. E04LBF will then terminate immediately with IFAIL set to the user's setting of IFLAG.

2: N – INTEGER *Input*

On entry: the number n of variables.

3: XC(N) – ***real*** array *Input*

On entry: the point x at which F and the $\dfrac{\partial F}{\partial x_j}$ are required.

4: FC – ***real*** *Output*

On exit: unless IFLAG is reset, FUNCT must set FC to the value of the objective function F at the current point x.

5: GC(N) – ***real*** array *Output*

On exit: unless IFLAG is reset, FUNCT must set GC(j) to the value of the first derivative $\dfrac{\partial F}{\partial x_j}$ at the point x, for $j = 1, 2, \ldots, n$.

6: IW(LIW) – INTEGER array *Workspace*
7: LIW – INTEGER *Input*
8: W(LW) – ***real*** array *Workspace*
9: LW – INTEGER *Input*

FUNCT is called with the same parameters IW, LIW, W and LW as for E04LBF. They are present so that, when other library routines require the solution of a minimization subproblem, constants needed for the function evaluation can be passed through IW and W. Similarly, users **could** use elements $3, 4, \ldots,$ LIW of IW and elements from $\max(8, 7 \times \mathrm{N} + \mathrm{N} \times (\mathrm{N} - 1)/2) + 1$ onwards of W for passing quantities to FUNCT from the (sub)program which calls E04LBF. However, because of the danger of mistakes in partitioning, it is recommended that users should pass information to FUNCT via COMMON and not use IW or W at all. In any case FUNCT **must not change** the first 2 elements of IW or the first $\max(8, 7 \times \mathrm{N} + \mathrm{N} \times (\mathrm{N} - 1)/2)$ elements of W.

FUNCT must be declared as EXTERNAL in the (sub)program from which E04LBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

Note: FUNCT should be tested separately before being used in conjunction with E04LBF.

3: HESS – SUBROUTINE, supplied by the user. *External Procedure*

HESS must calculate the second derivatives of F at any point x. (As with FUNCT, there is the option of causing E04LBF to terminate immediately.)

Its specification is:

```
SUBROUTINE HESS(IFLAG, N, XC, FHESL, LH, FHESD, IW, LIW, W, LW)
INTEGER          IFLAG, N, LH, IW(LIW), LIW, LW
real             XC(N), FHESL(LH), FHESD(N), W(LW)
```

1: IFLAG – INTEGER *Input/Output*

On entry: IFLAG is set to a non-negative number.

On exit: if HESS resets IFLAG to some negative number, E04LBF will terminate immediately with IFAIL set to the user's setting of IFLAG.

2: N – INTEGER *Input*

On entry: the number n of variables.

3: XC(N) – ***real*** array *Input*

On entry: the point x at which the second derivatives of F are required.

4: FHESL(LH) – ***real*** array *Output*

On exit: unless IFLAG is reset, HESS must place the strict lower triangle of the second derivative matrix of F (evaluated at the point x) in FHESL, stored by rows, i.e., set FHESL$((i-1)(i-2)/2+j) = \left.\dfrac{\partial^2 F}{\partial x_i \partial x_j}\right|_{\mathrm{XC}}$, for $i = 2, 3, \ldots, n$; $j = 1, 2, \ldots, i-1$. (The upper triangle is not required because the matrix is symmetric.)

5: LH – INTEGER *Input*

On entry: the length of the array FHESL.

6: FHESD(N) – ***real*** array *Input/Output*

On entry: the value of $\dfrac{\partial F}{\partial x_j}$ at the point x, for $j = 1, 2, \ldots, n$.

These values may be useful in the evaluation of the second derivatives.

On exit: unless IFLAG is reset, HESS must place the diagonal elements of the second derivative matrix of F (evaluated at the point x) in FHESD, i.e., set FHESD$(j) = \left.\dfrac{\partial^2 F}{\partial x_j^2}\right|_{\mathrm{XC}}$, $j = 1, 2, \ldots, n$.

7: IW(LIW) – INTEGER array *Workspace*
8: LIW – INTEGER *Input*
9: W(LW) – ***real*** array *Workspace*
10: LW – INTEGER *Input*

As in FUNCT, these parameters correspond to the parameters IW, LIW, W, LW of E04LBF. HESS **must not change** the first two elements of IW or the first $\max(8, 7 \times \mathrm{N} + \mathrm{N} \times (\mathrm{N}-1)/2)$ elements of W. Again, it is recommended that the user should pass quantities to HESS via COMMON and not use IW or W at all.

HESS must be declared as EXTERNAL in the (sub)program from which E04LBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

Note: HESS should be tested separately before being used in conjunction with E04LBF.

4: MONIT – SUBROUTINE, supplied by the user. *External Procedure*

If IPRINT ≥ 0, the user must supply a subroutine MONIT which is suitable for monitoring the minimization process. MONIT must not change the values of any of its parameters. If IPRINT < 0, a routine MONIT with the correct parameter list should still be supplied, although it will not be called.

Its specification is:

```
      SUBROUTINE MONIT(N, XC, FC, GC, ISTATE, GPJNRM, COND, POSDEF, NITER,
     1                 NF, IW, LIW, W, LW)
      INTEGER          N, ISTATE(N), NITER, NF, IW(LIW), LIW, LW
      real             XC(N), FC, GC(N), GPJNRM, COND, W(LW)
      LOGICAL          POSDEF
```

1: N – INTEGER *Input*

On entry: the number n of variables.

2: XC(N) – ***real*** array *Input*

On entry: the co-ordinates of the current point x.

3: FC – ***real*** *Input*

On entry: the value of $F(x)$ at the current point x.

4: GC(N) – ***real*** array *Input*

On entry: the value of $\dfrac{\partial F}{\partial x_j}$ at the current point x, for $j = 1, 2, \ldots, n$.

5: ISTATE(N) – INTEGER array *Input*

On entry: information about which variables are currently fixed on their bounds and which are free.

If ISTATE(j) is negative, x_j is currently:

fixed on its upper bound if ISTATE(j) $= -1$;

fixed on its lower bound if ISTATE(j) $= -2$;

effectively a constant (i.e., $l_j = u_j$) if ISTATE(j) $= -3$.

If ISTATE is positive, its value gives the position of x_j in the sequence of free variables.

6: GPJNRM – ***real*** *Input*

On entry: the Euclidean norm of the projected gradient vector g_z.

7: COND – ***real*** *Input*

On entry: the ratio of the largest to the smallest elements of the diagonal factor D of the projected Hessian matrix (see specification of HESS below). This quantity is usually a good estimate of the condition number of the projected Hessian matrix. (If no variables are currently free, COND is set to zero.)

8: POSDEF – LOGICAL *Input*

On entry: POSDEF is set .TRUE. or .FALSE. according to whether the second derivative matrix for the current subspace, H, is positive-definite or not.

9: NITER – INTEGER *Input*

On entry: the number of iterations (as outlined in Section 3) which have been performed by E04LBF so far.

10: NF – INTEGER *Input*

On entry: the number of times that FUNCT has been called so far. Thus NF is the number of function and gradient evaluations made so far.

11: IW(LIW) – INTEGER array *Workspace*
12: LIW – INTEGER *Input*
13: W(LW) – ***real*** array *Workspace*
14: LW – INTEGER *Input*

As in FUNCT, and HESS, these parameters correspond to the parameters IW, LIW, W, LW of E04LBF. They are included in MONIT's parameter list primarily for when E04LBF is called by other library routines.

MONIT must be declared as EXTERNAL in the (sub)program from which E04LBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

The user should normally print out FC, GPJNRM and COND so as to be able to compare the quantities mentioned in Section 7. It is normally helpful to examine XC, POSDEF and NF as well.

5: IPRINT – INTEGER *Input*

On entry: the frequency with which MONIT is to be called. If IPRINT > 0, MONIT is called once every IPRINT iterations and just before exit from E04LBF. If IPRINT $= 0$, MONIT is just called at the final point. If IPRINT < 0, MONIT is not called at all.

IPRINT should normally be set to a small positive number.

Suggested value: IPRINT $= 1$.

6: MAXCAL – INTEGER *Input*

On entry: the maximum permitted number of evaluations of $F(x)$, i.e., the maximum permitted number of calls of FUNCT.

Suggested value: MAXCAL $= 50 \times$ N.

Constraint: MAXCAL ≥ 1.

7: ETA – ***real*** *Input*

On entry: every iteration of E04LBF involves a linear minimization (i.e., minimization of $F(x + \alpha p)$ with respect to α). ETA specifies how accurately these linear minimizations are to be performed. The minimum with respect to α will be located more accurately for small values of ETA (say 0.01) than for large values (say 0.9).

Although accurate linear minimizations will generally reduce the number of iterations of E04LBF, this usually results in an increase in the number of function and gradient evaluations required for each iteration. On balance, it is usually more efficient to perform a low accuracy linear minimization.

Suggested value: **ETA $= 0.9$ is usually a good choice** although a smaller value may be warranted if the matrix of second derivatives is expensive to compute compared with the function and first derivatives.

If N $= 1$, ETA should be set to 0.0 (also when the problem is effectively 1-dimensional even though $n > 1$; i.e., if for all except one of the variables the lower and upper bounds are equal).

Constraint: $0.0 \leq$ ETA < 1.0.

8: XTOL – ***real*** *Input*

On entry: the accuracy in x to which the solution is required.

If x_{true} is the true value of x at the minimum, then x_{sol}, the estimated position prior to a normal exit, is such that $\|x_{sol} - x_{true}\| <$ XTOL $\times (1.0 + \|x_{true}\|)$, where $\|y\| = \sqrt{\sum_{j=1}^{n} y_j{}^2}$. For example, if the elements of x_{sol} are not much larger than 1.0 in modulus, and if XTOL is set to 10^{-5} then x_{sol} is usually accurate to about 5 decimal places. (For further details see Section 7.)

If the problem is scaled roughly as described in Section 8 and ϵ is the ***machine precision***, then $\sqrt{\epsilon}$ is probably the smallest reasonable choice for XTOL. (This is because, normally, to machine accuracy, $F(x + \sqrt{\epsilon}\, e_j) = F(x)$ where e_j is any column of the identity matrix.)

If the user sets XTOL to 0.0 (or any positive value less than ϵ), E04LBF will use $10.0 \times \sqrt{\epsilon}$ instead of XTOL.

Suggested value: XTOL $= 0.0$.

Constraint: XTOL ≥ 0.0.

9: STEPMX – ***real*** *Input*

On entry: an estimate of the Euclidean distance between the solution and the starting point supplied by the user. (For maximum efficiency a slight overestimate is preferable.)

E04LBF will ensure that, for each iteration,

$$\sqrt{\sum_{j=1}^{n}\left[x_j^{(k)} - x_j^{(k-1)}\right]^2} \leq \text{STEPMX}$$

where k is the iteration number. Thus, if the problem has more than one solution, E04LBF is most likely to find the one nearest to the starting point. On difficult problems, a realistic choice can prevent the sequence of $x^{(k)}$ entering a region where the problem is ill-behaved and can also help to avoid possible overflow in the evaluation of $F(x)$. However, an underestimate of STEPMX can lead to inefficiency.

Suggested value: STEPMX = 100000.0.

Constraint: STEPMX $\geq$ XTOL.

10: IBOUND – INTEGER *Input*

On entry: specifies whether the problem is unconstrained or bounded. If there are bounds on the variables, IBOUND can be used to indicate whether the facility for dealing with bounds of special forms is to be used. It must be set to one of the following values:

IBOUND = 0

If the variables are bounded and the user will be supplying all the l_j and u_j individually.

IBOUND = 1

If the problem is unconstrained.

IBOUND = 2

If the variables are bounded, but all the bounds are of the form $0 \leq x_j$.

IBOUND = 3

If all the variables are bounded, and $l_1 = l_2 = \cdots = l_n$ and $u_1 = u_2 = \cdots = u_n$.

IBOUND = 4

If the problem is unconstrained. (The IBOUND = 4 option is provided purely for consistency with other routines. In E04LBF it produces the same effect as IBOUND = 1.)

Constraint: $0 \leq$ IBOUND ≤ 4.

11: BL(N) – ***real*** array *Input/Output*

On entry: the fixed lower bounds l_j.

If IBOUND is set to 0, the user must set BL(j) to l_j, for $j = 1, 2, \ldots, n$. (If a lower bound is not specified for any x_j, the corresponding BL(j) should be set to a large negative number, e.g., -10^6.)

If IBOUND is set to 3, the user must set BL(1) to l_1; E04LBF will then set the remaining elements of BL equal to BL(1).

If IBOUND is set to 1, 2 or 4, BL will be initialised by E04LBF.

On exit: the lower bounds actually used by E04LBF, e.g., if IBOUND = 2, BL(1) = BL(2) = $\cdots$ = BL(N) = 0.0.

12: BU(N) – ***real*** array *Input/Output*

On entry: the fixed upper bounds u_j.

If IBOUND is set to 0, the user must set BU(j) to u_j, for $j = 1, 2, \ldots, n$. (If an upper bound is not specified for any variable, the corresponding BU(j) should be set to a large positive number, e.g., 10^6.)

If IBOUND is set to 3, the user must set BU(1) to u_1; E04LBF will then set the remaining elements of BU equal to BU(1).

If IBOUND is set to 1, 2 or 4, BU will then be initialised by E04LBF.

On exit: the upper bounds actually used by E04LBF, e.g., if IBOUND $= 2$, BU(1) $=$ BU(2) $= \cdots =$ BU(N) $= 10^6$.

13: X(N) – ***real*** array *Input/Output*

On entry: X(j) must be set to a guess at the jth component of the position of minimum, for $j = 1, 2, \ldots, n$.

On exit: the final point $x^{(k)}$. Thus, if IFAIL $= 0$ on exit, X(j) is the jth component of the estimated position of the minimum.

14: HESL(LH) – ***real*** array *Output*

See description of HESD below.

15: LH – INTEGER *Input*

On entry: the actual length of HESL as declared in the (sub)program from which E04LBF is called.

Constraint: LH $\geq \max(\text{N} \times (\text{N} - 1)/2, 1)$.

16: HESD(N) – ***real*** array *Output*

On exit: during the determination of a direction p_z (see Section 3), $H + E$ is decomposed into the product LDL^T, where L is a unit lower triangular matrix and D is a diagonal matrix. (The matrices H, E, L and D are all of dimension n_z, where n_z is the number of variables free from their bounds. H consists of those rows and columns of the full second derivative matrix which relate to free variables. E is chosen so that $H + E$ is positive-definite.)

HESL and HESD are used to store the factors L and D. The elements of the strict lower triangle of L are stored row by row in the first $n_z(n_z - 1)/2$ positions of HESL. The diagonal elements of D are stored in the first n_z positions of HESD.

In the last factorization before a normal exit, the matrix E will be zero, so that HESL and HESD will contain, on exit, the factors of the final second derivative matrix H. The elements of HESD are useful for deciding whether to accept the result produced by E04LBF (see Section 7).

17: ISTATE(N) – INTEGER array *Output*

On exit: information about which variables are currently on their bounds and which are free. If ISTATE(j) is:

equal to -1, x_j is fixed on its upper bound

equal to -2, x_j is fixed on its lower bound

equal to -3, x_j is effectively a constant (i.e., $l_j = u_j$)

positive, ISTATE(j) gives the position of x_j in the sequence of free variables.

18: F – ***real*** *Output*

On exit: the function value at the final point given in X.

19: G(N) – ***real*** array *Output*

On exit: the first derivative vector corresponding to the final point given in X. The components of G corresponding to free variables should normally be close to zero.

20: IW(LIW) – INTEGER array *Workspace*
21: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04LBF is called.

Constraint: LIW ≥ 2.

22: W(LW) – ***real*** array *Workspace*
23: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04LBF is called.

Constraint: LW $\geq \max(7 \times \mathrm{N} + \mathrm{N} \times (\mathrm{N}-1)/2, 8)$.

24: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output parameters may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL < 0

A negative value of IFAIL indicates an exit from E04LBF because the user has set IFLAG negative in FUNCT or HESS. The value of IFAIL will be the same as the user's setting of IFLAG.

IFAIL $= 1$

On entry, N < 1,
or MAXCAL < 1,
or ETA < 0.0,
or ETA ≥ 1.0,
or XTOL < 0.0,
or STEPMX $<$ XTOL,
or IBOUND < 0,
or IBOUND > 4,
or BL(j) $>$ BU(j) for some j if IBOUND $= 0$,
or BL(1) $>$ BU(1) if IBOUND $= 3$,
or LH $< \max(1, \mathrm{N} \times (\mathrm{N}-1)/2)$,
or LIW < 2,
or LW $< \max(8, 7 \times \mathrm{N} + \mathrm{N} \times (\mathrm{N}-1)/2)$.

(Note that if the user has set XTOL to 0.0, E04LBF uses the default value and continues without failing.) When this exit occurs no values will have been assigned to F or to the elements of HESL, HESD or G.

IFAIL $= 2$

There have been MAXCAL function evaluations. If steady reductions in $F(x)$ were monitored up to the point where this exit occurred, then the exit probably occurred simply because MAXCAL was set too small, so the calculations should be restarted from the final point held in x. This exit may also indicate that $F(x)$ has no minimum.

IFAIL $= 3$

The conditions for a minimum have not all been met, but a lower point could not be found.

Provided that, on exit, the first derivatives of $F(x)$ with respect to the free variables are sufficiently small, and that the estimated condition number of the second derivative matrix is not too large, this error exit may simply mean that, although it has not been possible to satisfy the specified requirements, the algorithm has in fact found the minimum as far as the accuracy of the machine permits. Such a situation can arise, for instance, if XTOL has been set so small that rounding errors in the evaluation of $F(x)$ or its derivatives make it impossible to satisfy the convergence conditions.

If the estimated condition number of the second derivative matrix at the final point is large, it could be that the final point is a minimum, but that the smallest eigenvalue of the Hessian matrix is so close to zero that it is not possible to recognise the point as a minimum.

IFAIL $= 4$

Not used. (This is done to make the significance of IFAIL $= 5$ similar for E04KDF and E04LBF.)

IFAIL $= 5$

All the Lagrange-multiplier estimates which are not indisputably positive lie relatively close to zero, but it is impossible either to continue minimizing on the current subspace or to find a feasible lower point by releasing and perturbing any of the fixed variables. The user should investigate as for IFAIL $= 3$.

The values IFAIL $= 2$, 3 and 5 may also be caused by mistakes in FUNCT or HESS, by the formulation of the problem or by an awkward function. If there are no such mistakes, it is worth restarting the calculations from a different starting point (not the point at which the failure occurred) in order to avoid the region which caused the failure.

7 Accuracy

A successful exit (IFAIL $= 0$) is made from E04LBF when $H^{(k)}$ is positive-definite and when (B1, B2 and B3) or B4 hold, where

$$\begin{aligned} \text{B1} &\equiv \alpha^{(k)} \times \|p^{(k)}\| < (\text{XTOL} + \sqrt{\epsilon}) \times (1.0 + \|x^{(k)}\|) \\ \text{B2} &\equiv |F^{(k)} - F^{(k-1)}| < (\text{XTOL}^2 + \epsilon) \times (1.0 + |F^{(k)}|) \\ \text{B3} &\equiv \|g_z^{(k)}\| < (\epsilon^{1/3} + \text{XTOL}) \times (1.0 + |F^{(k)}|) \\ \text{B4} &\equiv \|g_z^{(k)}\| < 0.01 \times \sqrt{\epsilon}. \end{aligned}$$

(Quantities with superscript k are the values at the kth iteration of the quantities mentioned in Section 3. ϵ is the ***machine precision*** and $\|.\|$ denotes the Euclidean norm.)

If IFAIL $= 0$, then the vector in X on exit, x_{sol}, is almost certainly an estimate of the position of the minimum, x_{true}, to the accuracy specified by XTOL.

If IFAIL $= 3$ or 5, x_{sol} may still be a good estimate of x_{true}, but the following checks should be made. Let the largest of the first n_z elements of HESD be HESD(b), let the smallest be HESD(s), and define $k =$ HESD(b)/HESD(s). The scalar k is usually a good estimate of the condition number of the projected Hessian matrix at x_{sol}. If

(i) the sequence $\{F(x^{(k)})\}$ converges to $F(x_{sol})$ at a superlinear or fast linear rate,

(ii) $\|g_z(x_{sol})\|^2 < 10.0 \times \epsilon$, and

(iii) $k < 1.0/\|g_z(x_{sol})\|$,

then it is almost certain that x_{sol} is a close approximation to the position of a minimum. When (ii) is true, then usually $F(x_{sol})$ is a close approximation to $F(x_{true})$. The quantities needed for these checks are all available via MONIT; in particular the value of COND in the last call of MONIT before exit gives k.

Further suggestions about confirmation of a computed solution are given in the E04 Chapter Introduction.

8 Further Comments

8.1 Timing

The number of iterations required depends on the number of variables, the behaviour of $F(x)$, the accuracy demanded and the distance of the starting point from the solution. The number of multiplications performed in an iteration of E04LBF is $\frac{n_z^3}{6} + \mathrm{O}(n_z^2)$. In addition, each iteration makes one call of HESS and at least one call of FUNCT. So, unless $F(x)$ and its derivatives can be evaluated very quickly, the run time will be dominated by the time spent in FUNCT and HESS.

8.2 Scaling

Ideally, the problem should be scaled so that, at the solution, $F(x)$ and the corresponding values of the x_j are each in the range $(-1, +1)$, and so that at points one unit away from the solution, $F(x)$ differs from its value at the solution by approximately one unit. This will usually imply that the Hessian matrix at the solution is well-conditioned. It is unlikely that the user will be able to follow these recommendations very closely, but it is worth trying (by guesswork), as sensible scaling will reduce the difficulty of the minimization problem, so that E04LBF will take less computer time.

8.3 Unconstrained Minimization

If a problem is genuinely unconstrained and has been scaled sensibly, the following points apply:

(a) n_z will always be n,

(b) HESL and HESD will be factors of the full second derivative matrix with elements stored in the natural order,

(c) the elements of g should all be close to zero at the final point,

(d) the values of the ISTATE(j) given by MONIT and on exit from E04LBF are unlikely to be of interest (unless they are negative, which would indicate that the modulus of one of the x_j has reached 10^6 for some reason),

(e) MONIT's parameter GPJNRM simply gives the norm of the first derivative vector.

So the following routine (in which partitions of extended workspace arrays are used as BL, BU and ISTATE) could be used for unconstrained problems:

```
      SUBROUTINE UNCLBF(N,FUNCT,HESS,MONIT,IPRINT,MAXCAL,ETA,XTOL,
     *                  STEPMX,X,HESL,LH,HESD,F,G,IWORK,LIWORK,WORK,
     *                  LWORK,IFAIL)
C
C     A ROUTINE TO APPLY E04LBF TO UNCONSTRAINED PROBLEMS.
C
C     THE REAL ARRAY WORK MUST BE OF DIMENSION AT LEAST
C     (9*N + MAX(1, N*(N-1)/2)).  ITS FIRST 7*N + MAX(1, N*(N-1)/2)
C     ELEMENTS WILL BE USED BY E04LBF AS THE ARRAY W.  ITS LAST
C     2*N ELEMENTS WILL BE USED AS THE ARRAYS BL AND BU.
C
C     THE INTEGER ARRAY IWORK MUST BE OF DIMENSION AT LEAST (N+2)
C     ITS FIRST 2 ELEMENTS WILL BE USED BY E04LBF AS THE ARRAY IW.
C     ITS LAST N ELEMENTS WILL BE USED AS THE ARRAY ISTATE.
C
C     LIWORK AND LWORK MUST BE SET TO THE ACTUAL LENGTHS OF IWORK
C     AND WORK RESPECTIVELY, AS DECLARED IN THE CALLING SEGMENT.
C
C     OTHER PARAMETERS ARE AS FOR E04LBF.
C
```

```
C     .. Parameters ..
      INTEGER NOUT
      PARAMETER (NOUT=6)
C     .. Scalar Arguments ..
      real ETA, F, STEPMX, XTOL
      INTEGER IFAIL, IPRINT, LH, LIWORK, LWORK, MAXCAL, N
C     .. Array Arguments ..
      real G(N), HESD(N), HESL(LH), WORK(LWORK), X(N)
      INTEGER IWORK(LIWORK)
C     .. Subroutine Arguments ..
      EXTERNAL FUNCT, HESS, MONIT
C     .. Local Scalars ..
      INTEGER IBOUND, J, JBL, JBU, NH
      LOGICAL TOOBIG
C     .. External Subroutines ..
      EXTERNAL E04LBF
C     .. Executable Statements ..
C     CHECK THAT SUFFICIENT WORKSPACE HAS BEEN SUPPLIED
      NH = N*(N-1)/2
      IF (NH.EQ.0) NH = 1
      IF (LWORK.LT.9*N+NH .OR.  LIWORK.LT.N+2) THEN
      WRITE (NOUT,FMT=99999)
      STOP
      END IF
C     JBL AND JBU SPECIFY THE PARTS OF WORK USED AS BL AND BU
      JBL = 7*N + NH + 1
      JBU = JBL + N
C     SPECIFY THAT THE PROBLEM IS UNCONSTRAINED
      IBOUND = 4
      CALL E04LBF(N,FUNCT,HESS,MONIT,IPRINT,MAXCAL,ETA,XTOL,STEPMX,
     * IBOUND,WORK(JBL),WORK(JBU),X,HESL,LH,HESD,IWORK(3),F,
     * G,IWORK,LIWORK,WORK,LWORK,IFAIL)
C     CHECK THE PART OF IWORK WHICH WAS USED AS ISTATE IN CASE
C     THE MODULUS OF SOME X(J) HAS REACHED E+6
      TOOBIG = .FALSE.
      DO 20 J = 1, N
         IF (IWORK(2+J).LT.0) TOOBIG = .TRUE.
   20 CONTINUE
      IF ( .NOT.  TOOBIG) RETURN
      WRITE (NOUT,FMT=99998)
      STOP
C
99999 FORMAT (' ***** INSUFFICIENT WORKSPACE HAS BEEN SUPPLIED *****')
99998 FORMAT (' ***** A VARIABLE HAS REACHED E+6 IN MODULUS - NO UNCON',
     *        'STRAINED MINIMUM HAS BEEN FOUND *****')
      END
```

9 Example

A program to minimize

$$F = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

subject to the bounds

$$\begin{aligned} 1 &\leq x_1 \leq 3 \\ -2 &\leq x_2 \leq 0 \\ 1 &\leq x_4 \leq 3. \end{aligned}$$

starting from the initial guess (3, −1, 0, 1). Before calling E04LBF, the program calls E04HCF and E04HDF to check the derivatives calculated by FUNCT and HESS.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     E04LBF Example Program Text.
```

```
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, LH, LIW, LW
      PARAMETER        (N=4,LH=N*(N-1)/2,LIW=2,LW=7*N+N*(N-1)/2)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             ETA, F, STEPMX, XTOL
      INTEGER          IBOUND, IFAIL, IPRINT, J, MAXCAL
*     .. Local Arrays ..
      real             BL(N), BU(N), G(N), HESD(N), HESL(LH), W(LW),
     +                 X(N)
      INTEGER          ISTATE(N), IW(LIW)
*     .. External Subroutines ..
      EXTERNAL         E04HCF, E04HDF, E04LBF, FUNCT, HESS, MONIT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04LBF Example Program Results'
*     Set up an arbitrary point at which to check the derivatives
      X(1) = 1.46e0
      X(2) = -0.82e0
      X(3) = 0.57e0
      X(4) = 1.21e0
*     Check the 1st derivatives
      IFAIL = 0
*
      CALL E04HCF(N,FUNCT,X,F,G,IW,LIW,W,LW,IFAIL)
*
*     Check the 2nd derivatives
      IFAIL = 0
*
      CALL E04HDF(N,FUNCT,HESS,X,G,HESL,LH,HESD,IW,LIW,W,LW,IFAIL)
*
*     Continue setting parameters for E04LBF
*     * Set IPRINT to 1 to obtain output from MONIT at each iteration *
      IPRINT = -1
      MAXCAL = 50*N
      ETA = 0.9e0
*     Set XTOL to zero so that E04LBF will use the default tolerance
      XTOL = 0.0e0
*     We estimate that the minimum will be within 4 units of the
*     starting point
      STEPMX = 4.0e0
      IBOUND = 0
      BL(1) = 1.0e0
      BU(1) = 3.0e0
      BL(2) = -2.0e0
      BU(2) = 0.0e0
*     X(3) is not bounded, so we set BL(3) to a large negative
*     number and BU(3) to a large positive number
      BL(3) = -1.0e6
      BU(3) = 1.0e6
      BL(4) = 1.0e0
      BU(4) = 3.0e0
*     Set up starting point
      X(1) = 3.0e0
      X(2) = -1.0e0
      X(3) = 0.0e0
      X(4) = 1.0e0
      IFAIL = 1
*
      CALL E04LBF(N,FUNCT,HESS,MONIT,IPRINT,MAXCAL,ETA,XTOL,STEPMX,
     +            IBOUND,BL,BU,X,HESL,LH,HESD,ISTATE,F,G,IW,LIW,W,LW,
     +            IFAIL)
*
      IF (IFAIL.NE.0) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Error exit type', IFAIL,
     +     ' - see routine document'
      END IF
      IF (IFAIL.NE.1) THEN
         WRITE (NOUT,*)
```

```
         WRITE (NOUT,99998) 'Function value on exit is ', F
         WRITE (NOUT,99997) 'at the point', (X(J),J=1,N)
         WRITE (NOUT,*)
     +     'The corresponding (machine dependent) gradient is'
         WRITE (NOUT,99996) (G(J),J=1,N)
         WRITE (NOUT,99995) 'ISTATE contains', (ISTATE(J),J=1,N)
         WRITE (NOUT,99994) 'and HESD contains', (HESD(J),J=1,N)
       END IF
       STOP
*
99999 FORMAT (1X,A,I3,A)
99998 FORMAT (1X,A,F9.4)
99997 FORMAT (1X,A,4F9.4)
99996 FORMAT (23X,1P,4e12.3)
99995 FORMAT (1X,A,4I5)
99994 FORMAT (1X,A,4e12.4)
      END
*
      SUBROUTINE FUNCT(IFLAG,N,XC,FC,GC,IW,LIW,W,LW)
*     Routine to evaluate objective function and its 1st derivatives.
*     .. Scalar Arguments ..
      real              FC
      INTEGER           IFLAG, LIW, LW, N
*     .. Array Arguments ..
      real              GC(N), W(LW), XC(N)
      INTEGER           IW(LIW)
*     .. Executable Statements ..
      FC = (XC(1)+10.0e0*XC(2))**2 + 5.0e0*(XC(3)-XC(4))**2 + (XC(2)
     +     -2.0e0*XC(3))**4 + 10.0e0*(XC(1)-XC(4))**4
      GC(1) = 2.0e0*(XC(1)+10.0e0*XC(2)) + 40.0e0*(XC(1)-XC(4))**3
      GC(2) = 20.0e0*(XC(1)+10.0e0*XC(2)) + 4.0e0*(XC(2)-2.0e0*XC(3))**3
      GC(3) = 10.0e0*(XC(3)-XC(4)) - 8.0e0*(XC(2)-2.0e0*XC(3))**3
      GC(4) = 10.0e0*(XC(4)-XC(3)) - 40.0e0*(XC(1)-XC(4))**3
      RETURN
      END
*
      SUBROUTINE HESS(IFLAG,N,XC,FHESL,LH,FHESD,IW,LIW,W,LW)
*     Routine to evaluate 2nd derivatives
*     .. Scalar Arguments ..
      INTEGER          IFLAG, LH, LIW, LW, N
*     .. Array Arguments ..
      real              FHESD(N), FHESL(LH), W(LW), XC(N)
      INTEGER          IW(LIW)
*     .. Executable Statements ..
      FHESD(1) = 2.0e0 + 120.0e0*(XC(1)-XC(4))**2
      FHESD(2) = 200.0e0 + 12.0e0*(XC(2)-2.0e0*XC(3))**2
      FHESD(3) = 10.0e0 + 48.0e0*(XC(2)-2.0e0*XC(3))**2
      FHESD(4) = 10.0e0 + 120.0e0*(XC(1)-XC(4))**2
      FHESL(1) = 20.0e0
      FHESL(2) = 0.0e0
      FHESL(3) = -24.0e0*(XC(2)-2.0e0*XC(3))**2
      FHESL(4) = -120.0e0*(XC(1)-XC(4))**2
      FHESL(5) = 0.0e0
      FHESL(6) = -10.0e0
      RETURN
      END
*
      SUBROUTINE MONIT(N,XC,FC,GC,ISTATE,GPJNRM,COND,POSDEF,NITER,NF,IW,
     +                 LIW,W,LW)
*     Monitoring routine
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalar Arguments ..
      real              COND, FC, GPJNRM
      INTEGER          LIW, LW, N, NF, NITER
      LOGICAL          POSDEF
*     .. Array Arguments ..
      real              GC(N), W(LW), XC(N)
      INTEGER          ISTATE(N), IW(LIW)
*     .. Local Scalars ..
```

```
      INTEGER          ISJ, J
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
     +' Itn     Fn evals              Fn value            Norm of proj g
     +radient'
      WRITE (NOUT,99999) NITER, NF, FC, GPJNRM
      WRITE (NOUT,*)
      WRITE (NOUT,*)
     +  ' J           X(J)               G(J)           Status'
      DO 20 J = 1, N
         ISJ = ISTATE(J)
         IF (ISJ.GT.0) THEN
            WRITE (NOUT,99998) J, XC(J), GC(J), '    Free'
         ELSE IF (ISJ.EQ.-1) THEN
            WRITE (NOUT,99998) J, XC(J), GC(J), '    Upper Bound'
         ELSE IF (ISJ.EQ.-2) THEN
            WRITE (NOUT,99998) J, XC(J), GC(J), '    Lower Bound'
         ELSE IF (ISJ.EQ.-3) THEN
            WRITE (NOUT,99998) J, XC(J), GC(J), '    Constant'
         END IF
   20 CONTINUE
      IF (COND.NE.0.0e0) THEN
         IF (COND.GT.1.0e6) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*)
     +'Estimated condition number of projected Hessian is more than 1.0E
     ++6'
         ELSE
            WRITE (NOUT,*)
            WRITE (NOUT,99997)
     +        'Estimated condition number of projected Hessian = ', COND
         END IF
         IF ( .NOT. POSDEF) THEN
*           The following statement is included so that this MONIT
*           can also be used in conjunction with E04KDF
            WRITE (NOUT,*)
            WRITE (NOUT,*)
     +        'Projected Hessian matrix is not positive definite'
         END IF
         RETURN
      END IF
*
99999 FORMAT (1X,I3,6X,I5,2(6X,1P,e20.4))
99998 FORMAT (1X,I2,1X,1P,2e20.4,A)
99997 FORMAT (1X,A,1P,e10.2)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 E04LBF Example Program Results

 Error exit type  3 - see routine document

 Function value on exit is    2.4338
 at the point   1.0000  -0.0852   0.4093   1.0000
 The corresponding (machine dependent) gradient is
                        2.953E-01  -5.867E-10   1.173E-09   5.907E+00
 ISTATE contains   -2    1    2   -2
 and HESD contains  0.2098E+03  0.4738E+02  0.4552E+02  0.1750E+02
```